Proceedings of the 14th International Conference on Educational Data Mining

I-Han Sharon Hsiao, Shaghayegh Sherry Sahebi, François Bouchet, Jill-Jênn Vie

Proceedings of the 14^{th} International Conference on Educational Data Mining

I-Han (Sharon) Hsiao, Shaghayegh (Sherry) Sahebi, François Bouchet, Jill-Jênn Vie (eds).

Proceedings of the 14th International Conference on Educational Data Mining
I-Han (Sharon) Hsiao, Shaghayegh (Sherry) Sahebi, François Bouchet, Jill-Jênn Vie (eds).
June 29th – July 2nd 2021. Paris, France.
ISBN: 978-1-7336736-2-4

PREFACE

For this 14th iteration of the International Conference on Educational Data Mining (EDM 2021), the conference was held completely online. EDM is organized under the auspices of the International Educational Data Mining Society and was meant to happen in Paris, France. The conference, held June 29th through July 2nd, 2021, follows thirteen previous editions (Online, 2020, Montreal 2019, Buffalo 2018, Wuhan 2017, Raleigh 2016, Madrid 2015, London 2014, Memphis 2013, Chania 2012, Eindhoven 2011, Pittsburgh 2010, Córdoba 2009, and Montreal 2008).

The official theme of this year's conference was Shifting Landscape of Education: Improving Blended and Distance Learning. This theme focused on identifying learning or teaching strategies that can be used to improve learning in various formats, such as partially or fully online, synchronous or asynchronous, and centralized or federated. In addition to the general topics, we welcomed research in the following areas: receiving implicit and explicit feedback from learners in BDL environments, interacting with students to ensure no learner is left behind, integrating and utilizing learning analytics in BDL environments to cope with switching between in-person and online modes, and addressing emerging privacy and ethical challenges in the new learning setting. This year's conference featured three invited talks: Cristina Conati, Professor at University of British Columbia; Sidney D'Mello, Associate Professor at University of Colorado Boulder; and Pierre Dillenbourg, Professor at École Polytechnique Fédérale de Lausanne.

Building on the policy started in 2019, EDM 2021 continued using a double-blind review process. To continue EDM 2020 efforts, the conference's Program Committee was significantly expanded by inviting new committee members from the past authors of the EDM community and the PC members of the related conferences. In total, 33 senior and 74 ordinary program committee members, in addition to the conference and track chairs, contributed to the reviewing process. This year, we received a total of 100 full-paper submissions and 84 short-paper submissions. From the full-paper submissions, 22% were accepted as full papers, 25% were accepted as short papers, and 16% were accepted as posters. From the short-paper submissions, 23% were accepted as short papers and 23% were accepted as posters.

Review & Decision Processes: For transparency and possible benefit of future EDM conferences, we are providing a detailed description of the paper

review and decision processes for the Full and Short paper tracks at EDM 2021:

1. After all papers were submitted, the Program Committee (PC) and Senior Program Committee (SPC) members directly bid on which papers they would like to review.

2. If committee members did not bid on papers after several reminders, bids were assigned to them. This was done automatically via the EasyChair conference management system.

3. Given the PC and SPC bids, the Program Chairs assigned papers to reviewers using EasyChair's automatic assignment option. This assignment maximizes the total score of the assignment, with high weight on matches where the bid was a "yes", medium weight on matches where the bid was a "maybe", and low weight on matches where the bid was a "no". The assignments were checked by the program chairs to ensure review restrictions of the program committee members and the number of reviews they would have preferred, if they had any. Each paper was assigned to one SPC member and two PC members. Considering the increased number of submissions and the review limitations of PC members, each PC member received on average 5.2 papers, and each SPC member received on average 4.1 papers. The maximum number of papers assigned to a program committee member was 6 papers. The automated reviewing assignment was manually checked to ensure fairness to reviewers in being primarily assigned papers for which they had entered positive bids, fairness to papers in being primarily assigned reviewers who had bid positively on that paper, and that automatic conflict detection had accurately detected conflicts. A set of changes was made based on this manual check, either due to assigning a paper to all reviewers that had bid "no" on it or due to assigning multiple papers to a reviewer who had assigned a "no" bid on all of them.

4. In an effort to increase the mean and decrease the variance in review quality, the Program Chairs defined reviewing guidelines, both for the PC and the SPC. These guidelines were posted to the EDM 2021 website and also linked in emails sent to reviewers.

5. At the end of the review period, the Program Chairs identified papers that received fewer than 3 reviews, as well as papers whose reviews were clearly lacking (e.g., just 1-2 sentences). Emergency reviewers (including the Program Chairs) were identified, and papers were assigned to them.

6. The Program Chairs examined the meta-reviews and acceptance/rejection recommendations for all papers. For any papers lacking a meta-review, the Program Chairs read the reviews and the paper, wrote a meta-review, and arrived at a recommendation for acceptance/rejection.

7. Papers were ranked by their weighted average review scores. The Program Chairs then manually identified and examined papers in "critical regions" of the ranking in which there was large variance in the meta-reviewers' decision recommendations (Accept as Full, Accept as Short, Accept as Poster, Reject) or

there was large difference between the weighted and unweighted average review scores. The goal here was to ensure that, in the opinions of both Program Chairs, all papers accepted as either Full or Short exhibited sufficient rigor for publication as such. When in doubt, the more conservative outcome (i.e., Accept as Short rather than Full, or Accept as Poster rather than Short) was chosen. In particular:

(a) For the Full paper track, the following range was calculated: Let m_f be the lowest score of any paper recommended by its meta-reviewer for "Accept as full", and let n_s be the highest score of any paper recommended by its meta-reviewer for "Accept as short". For any paper recommended for "Accept as full" whose score was in $[m_f, n_s]$, the Program Chairs discussed the paper and decided jointly whether to Accept as Full or Short. This deliberation focused on the question: "Do the reviewers point out important methodological or other fundamental problems that could significantly threaten validity?"

(b) The analogous process (both for papers submitted as Full, and for papers submitted as Short) was applied to papers whose weighted or unweighted average review scores were in the range $[m_s, n_p]$, where m_s is the lowest score of any paper recommended for Accept as Short and n_p is the highest score of any paper recommended for Accept as Poster.

(c) All other papers – i.e., those whose unweighted average review scores were outside the ranges described above – were accepted/rejected according to the recommendation of their assigned meta-reviewer.

During all aspects of both the Review and Decision processes, no Program Chair examined or handled any paper on which she was a co-author; any such paper was seen and handled exclusively by the other Chair to avoid a conflict of interest. (No papers were co-authored by both Program Chairs.)

Note that papers submitted to the Industry, Doctoral Consortium, Poster/Demo, and Workshop components of EDM 2021 had their own reviewing processes that were defined by the corresponding chairs in consultation with the Program Chairs. Papers published in the Poster/Demo track are the union of those submitted & accepted as Posters/Demos, and those submitted as either the Full or Short tracks that were accepted as Posters.

Posters/Demos: In addition to the Full or Short paper submissions that were accepted as posters mentioned above, there was a dedicated Poster/Demo track to which papers could be submitted directly. This track accepted 10 contributions out of 20 submissions.

JEDM: Together with the Journal of Educational Data Mining (JEDM), the EDM 2021 conference held a JEDM Track that provides researchers a venue to

deliver more substantial mature work than is possible in a conference proceeding and to present their work to a live audience. The papers submitted to this track followed the JEDM peer review process. Three JEDM papers are featured in the conference's program.

Industry: The main conference invited contributions to an Industry Track in addition to the main track. The EDM 2021 Industry Track received 5 submissions, of which 4 were accepted.

Doctoral Consortium: The EDM conference continues its tradition of providing opportunities for young researchers to present their work and receive feedback from their peers and senior researchers. The doctoral consortium this year features 4 such presentations.

Paper Topics: In terms of topics of all submitted papers, the tables (Table 1 & Table 2) below list the top 20 popular keywords associated with papers created by the EasyChair system:

Keyword	**Weight**
Educational Data Mining	1000
Knowledge Tracing	624
Machine Learning	494
Neural network	426
Learning Analytics	397
Higher education	305
Student performance	299
Intelligent Tutoring System	284
Computer Science	278
Data Mining	259
Self-regulated learning	235
Deep Knowledge Tracing	209
Deep Learning	174
Natural Language Processing	167
Response time	164
Research question	161
Automated Essay Scoring	155
Artificial Intelligence	154
Peer assessment	149

Keyword	Weight
Learning environment	142

Table 1. Top 20 keywords associated with submitted papers

Also, the table below lists the top 20 popular keywords associated with the accepted papers:

Keyword	Weight
Educational Data Mining	300
Knowledge Tracing	163
Causal inference	124
Machine Learning	96
Neural network	90
Learning Analytics	88
Educational social network	74
Computer Science	70
Learning Agency	68
Hidden Markov Model	67
Intelligent Tutoring System	66
Knowledge State	66
Data Mining	65
Student performance	58
Stack Overflow	53
Student knowledge state	49
Computational linguistic	46
Problem solving	45
Curricular pattern	42
Student success	38

Table 2. Top 20 keywords associated with accepted papers

Best Paper, Presentation, and Reviewer Awards

Following the past EDM traditions, one best full paper, one best short paper,

and one best student paper were selected and awarded. The selection process included the program chairs reviewing the papers with praised and consistently high rated papers by the program committee and the recommended papers by the senior program committee members. After selecting four nominees from all the full and short papers, a committee of senior EDM members voted, met, and conferred to select the awardees. The best student paper was selected from the list of full paper nominees, since all of them has student first-authors. The list of best paper awardees are:

The best full paper:

Just a Few Expert Constraints Can Help: Humanizing Data-Driven Subgoal Detection for Novice Programming. By Samiha Marwan, Yang Shi, Ian Menezes, Min Chi, Tiffany Barnes and Thomas Price

The best student paper:

Early Prediction of Conceptual Understanding in Interactive Simulations. By Jade Cock, Mirko Marras, Christian Giang and Tanja Käser

The best short paper:

Do Common Educational Datasets contain Static Information? A Statistical Study. By Théo Barollet, Florent Bouchez-Tichadou and Fabrice Rastello

As the conference moved to the online setting, the program chairs decided to add best paper presentation and best poster presentation awards to encourage high-quality presentations by paper authors and engagement and attendance of the community. These awards were selected by the EDM conference community and attendees in a rank-based voting system. The attendees could select rank paper and poster presentations separately via two online Google forms, by selecting the best, second-best, and third-best presentations during the conference. To avoid memory availability bias, the forms were extensively advertised in the daily emails sent to the attendees by the general chairs, and at the end of each session. The votes were tallied before the closing session to announce the awardees.

The best poster presentation:

Are Violations of Student Privacy "Quick and Easy"? Investigating the Privacy of Students' Images and Names in the Context of K-12 Educational Institution's Posts on Facebook. By Macy Burchfield, Joshua Rosenberg, Conrad Borchers, Tayla Thomas, Benjamin Gibbons and Christian Fischer

The best paper presentation:

Early Prediction of Conceptual Understanding in Interactive Simulations. By Jade Cock, Mirko Marras, Christian Giang and Tanja Käser

In addition to the above, as a way to thank the current program committee members and to encourage serving as a program committee member and providing high-quality reviews in the future EDM conferences, the program chairs added the best reviewer award to the list of awards. These reviewers were selected

by the program chairs after carefully reading all the reviews and meta-reviews of all papers, focusing on the reviewers providing extraordinary reviews, such as suggestions on how to improve the authors' works and mentioning relevant literature to them, and the ones who have been on-time or volunteering to review more papers than average. The list of the best reviewers is:

Agathe Merceron, Beuth University of Applied Sciences Berlin

Andrew Olney, University of Memphis

Anna Rafferty, Carleton College

Cheng-Yu Chung, Arizona State University

Christopher Brooks, University of Michigan

Dragan Gasevic, Monash University

Giora Alexandron, Weizmann Institute of Science

James Lester, North Carolina State University

Joshua Gardner, University of Michigan

Julio Guerra, Universidad Austral de Chile

Sébastien Lallé, The University of British Columbia, Department of Computer Science

Sergey Sosnovsky, Utrecht University

Shalini Pandey, University of Minnesota

Stephen Fancsali, Carnegie Learning, Inc.

Tanja Käser, EPFL

Vincent Aleven, Human-Computer Interaction Institute, Carnegie Mellon University

Test of Time Award: Following in the footsteps of last year's conference, EDM 2021 also includes an invited talk by the authors of the 2020 winner of the EDM Test of Time Award. This year's talk is delivered by Cristóbal Romero.

Workshops: In addition to the main program, there are six workshops accepted, including Reinforcement Learning for Education: Opportunities and Challenges; Causal Inference in Educational Data Mining; Workshop for Undergraduates in Educational Data Mining and Learning Engineering; A Workshop on Process Analysis Methods For Educational Data; The Second Workshop of The Learner Data Institute: Big Data, Research Challenges, & Science Convergence in Educational Data Science; The 5th Educational Data for Mining in Computer Science Education (CSEDM).

Coronavirus: This year's conference was originally arranged to take place in Paris, France. Due to the SARS-CoV-2 (coronavirus) epidemic, EDM 2021, as

well as most other academic conferences in 2021, had to be changed to a purely online format. This presented some difficulties, especially of how to engage and encourage interaction among participants using just Zoom and other online tools (i.e. Gather Town, SpeakUp, Whova, etc.) rather than face-to-face meetings. However, it also significantly reduced the costs of conducting and attending the conference since physical meeting spaces, air travel, and on-site lodging were no longer necessary – and this arguably increased our conference's accessibility. To facilitate efficient transmission of presentations, especially given that not everyone's Internet connection could be guaranteed to be stable, we required all paper presenters to pre-record their presentation as a video and then to host it online.

Thanks: We thank Direction du numérique pour l'éducation (French MoE), Pix, CNRS, Inria, Eedi, EvidenceB, Educational Testing Service (ETS), Duolingo as sponsors of EDM 2021 for their generous support, especially during this financially difficult time of the coronavirus. We are also grateful to the individual conference chairs, the senior program committee, regular program committee members, subreviewers, emergency reviewers, and IEDMS board members, without whose expert input and hard work this conference would not be possible. Finally, we thank the entire organizing team and all authors who submitted their work to EDM 2021.

I-Han (Sharon) Hsiao	Santa Clara University, CA, USA	Program Chair
Shaghayegh (Sherry) Sahebi	University at Albany, NY, USA	Program Chair
François Bouchet	Sorbonne University, Paris, France	General Chair
Jill-Jênn Vie	Inria, France	General Chair

July 2nd, 2021

Organizing Committee

General Chairs:

- François Bouchet (Sorbonne University, Paris, France)
- Jill-Jênn Vie (Inria, France)

Program Chairs:

- I-Han (Sharon) Hsiao (Santa Clara University, CA, USA)
- Shaghayegh (Sherry) Sahebi (University at Albany, NY, USA)

Workshop & Tutorial Chairs:

- Thomas Price (North Carolina State University, NC, USA)

- Sweet San Pedro (ACT, Inc.)

Industry Track Chairs:

- Giora Alexandron (Weizmann Institute of Science, Israel)
- Niki Gitinabard (North Carolina State University, NC, USA)

Doctoral Consortium Chairs:

- Min Chi (North Carolina State University, NC, USA)
- Gautam Biswas (Vanderbilt University, TN, USA)

JEDM Track Chairs:

- Amal Zouaq (Ecole Polytechnique de Montréal, QC, Canada)
- Olga Santos (UNED, Spain)

Poster & Demo Track Chairs:

- Ange Tato (Université du Québec à Montréal, QC, Canada)
- Hassan Khosravi (University of Queensland, Australia)

Publication/Proceedings Chairs:

- Fatima Harrak (Sorbonne University, Paris, France)
- Cheng-Yu Chung (Arizona State University, AZ, USA)

Sponsorship Chair:

- Benoît Choffin (Paris-Saclay University, France)

Publicity/Social Media Chair:

- Khushboo Thaker (University of Pittsburgh, PA, USA)

Web Chair:

- Paul Salvador Inventado (California State University Fullerton, CA, USA)

IEDMS Officers:

Kenneth Koedinger, President, Carnegie Mellon University, USA

Mingyu Feng, Treasurer, WestEd, USA

IEDMS Board of Directors:

Ryan Baker, University of Pennsylvania, USA

Mykola Pechenizkiy, Eindhoven University of Technology, Netherlands

Michel Desmarais, École Polytechnique de Montréal, Canada

Tiffany Barnes, North Carolina State Universty, USA

Kalina Yacef, University of Sydney, Australia

Rakesh Agrawal, Data Insights Laboratories, USA

Luc Paquette, University of Illinois Urbana-Champaign, USA

Neil Heffernan, Worcester Polytechnic Institute, USA

Anna Rafferty, Carleton College, USA

Andrew Olney, JEDM Editor, University of Memphis, USA

John Stamper, Carnegie Mellon University, USA

Sidney D'Mello, University of Colorado Boulder, USA

Senior Program Committee Members

Name	Affiliation
Agathe Merceron	Beuth University of Applied Sciences Berlin
Alex Bowers	Columbia University
Andrew Olney	University of Memphis
Anna Rafferty	Carleton College
Bradford Mott	North Carolina State University
Collin Lynch	North Carolina State University
Cristóbal Romero	University of Córdoba
Dragan Gasevic	Monash University
James Lester	North Carolina State University
Jesus G. Boticario	UNED
John Stamper	Carnegie Mellon University
José González-Brenes	Chegg
Kenneth Koedinger	Carnegie Mellon University
Kristy Elizabeth Boyer	University of Florida
Luc Paquette	University of Illinois at Urbana-Champaign
Ma. Mercedes T. Rodrigo	Ateneo de Manila University
Martina Rau	University of Wisconsin - Madison
Min Chi	North Carolina State University
Mingyu Feng	WestEd
Neil Heffernan	Worcester Polytechnic Institute

Name	Affiliation
Niels Pinkwart	Humboldt-Universität zu Berlin
Noboru Matsuda	North Carolina State University
Philip I. Pavlik Jr.	University of Memphis
Radek Pelánek	Masaryk University Brno
Ryan Baker	University of Pennsylvania
Sebastián Ventura	University of Córdoba
Stefan Trausan-Matu	Politehnica University of Bucharest
Stephan Weibelzahl	Private University of Applied Sciences Göttingen
Stephen Fancsali	Carnegie Learning, Inc.
Steven Ritter	Carnegie Learning, Inc.
Vanda Luengo	Sorbonne University - LIP6
Vincent Aleven	Carnegie Mellon University
Zach Pardos	University of California, Berkeley

Program Committee Members

Name	Affiliation
Abelardo Pardo	University of South Australia
Alexandra Cristea	Durham University
Alfredo Zapata González	Universidad Autonoma de Yucatan
Amal Zouaq	Ecole Polytechnique de Montréal
Amelia Zafra Gómez	Universidad de Granada
Anthony F. Botelho	Worcester Polytechnic Institute
Armelle Brun	LORIA - Université de Lorraine
Benoît Choffin	EvidenceB
Burcu Arslan	Educational Testing Service
Caitlin Tenison	Educational Testing Service
Carol Forsyth	Educational Testing Service
Cheng-Yu Chung	Arizona State University
Christopher Brooks	University of Michigan

Name	Affiliation
Chunpai Wang	University at Albany - SUNY
Cristian Mihaescu	University of Craiova
David Pritchard	Massachusetts Institute of Technology
Diego Zapata-Rivera	Educational Testing Service
Donatella Merlini	Università di Firenze
Ella Haig	University of Portsmouth
Erik Hemberg	ALFA
Fatima Harrak	Sorbonne University - LIP6
Giora Alexandron	Weizmann Institute of Science
Guojing Zhou	University of Colorado Boulder
Hassan Khosravi	The University of Queensland
Irene-Angelica Chounta	University of Duisburg-Essen
Ivan Luković	University of Novi Sad
Ivon Arroyo	University of Massachusetts Amherst
Jacob Whitehill	Worcester Polytechnic Institute
Javier Bravo-Agapito	Madrid Open University (UDIMA)
Jeremiah Folsom-Kovarik	Soar Technology, Inc.
Jiangang Hao	Educational Testing Service
Jihyun Park	Apple, Inc.
Jina Kang	Utah State University
José Raúl Romero	University of Córdoba
Joshua Gardner	University of Michigan
Juan Alfonso Lara Torralbo	Open University of Madrid, UDIMA
Julio Guerra	Universidad Austral de Chile
Jun-Ming Su	National University of Tainan
Keith Brawner	United States Army Research Laboratory
Ling Tan	Australian Council for Educational Research
Marcelo Worsley	Northwestern University
Maria Luque	University of Córdoba
Mehmet Celepkolu	University of Florida
Mengfan Yao	University at Albany - SUNY

Name	Affiliation
Miguel Ángel Conde	University of León
Mirko Marras	École Polytechnique Fédérale de Lausanne - EPFL
Nicholas Diana	Colgate University
Nigel Bosch	University of Illinois Urbana-Champaign
Niki Gitinabard	North Carolina State University
Olga C. Santos	aDeNu Research Group (UNED)
Paul Salvador Inventado	California State University Fullerton
Paul Stefan Popescu	University of Craiova
Pedro J. Muñoz-Merino	Universidad Carlos III de Madrid
Rémi Venant	Le Mans Université - LIUM
Renza Campagni	Università degli Studi di Firenze
Scott Crossley	Georgia State University
Sébastien Lallé	The University of British Columbia
Sergey Sosnovsky	Utrecht University
Shahab Boumi	University of Central Florida
Shalini Pandey	University of Minnesota
Shayan Doroudi	Carnegie Mellon University
Siqian Zhao	University at Albany - SUNY
Sotiris Kotsiantis	University of Patras
Tanja Käser	École Polytechnique Fédérale de Lausanne - EPFL
Victor Menendez-Dominguez	Universidad Autónoma de Yucatán
Vladimir Ivančević	University of Novi Sad
Wookhee Min	North Carolina State University
Yancy Vance Paredes	Arizona State University
Yang Jiang	Educational Testing Service
Yolaine Bourda	LRI, CentraleSupélec

Subreviewers

Name	Affiliation
Aaron Alphonsus	Worcester Polytechnic Institute
Aaron Haim	Worcester Polytechnic Institute
Adam Gaweda	North Carolina State University
Alberto Jiménez Macías	Universidad Carlos III de Madrid
Ana Serrano Mamolar	UNED
Anjali Singh	University of Michigan
Anshul Aggarwal	University of Michigan
Ashish Gurung	Worcester Polytechnic Institute
Aubrey Condor	University of California, Berkeley
Aurora Ramírez	University of Córdoba
Benoît Choffin	EvidenceB
Cheng-Yu Chung	Arizona State University
David Martin Gomez	Universidad Carlos III de Madrid
Effat Farhana	North Carolina State University
Eliana Scheihing	Universidad Austral de Chile
Erzhuo Shao	Tsinghua University
Esha Sharma	North Carolina State University
Ethan Prihar	Worcester Polytechnic Institute
Fernando Rodriguez	University of Florida
Ge Gao	North Carolina State University
Guojing Zhou	North Carolina State University
Heeryung Choi	University of Michigan
Hyunwoo Sohn	North Carolina State University
Jialin Yu	Durham University
Jing Zhang	New York University
Joseph Wiggins	University of Florida
Khulood Alharbi	Durham University
Kimberly Michelle Ying	University of Florida
Machi Shimmei	North Carolina State University
Maria Cecilia Verri	Università di Firenze
Markel Sanz Ausin	North Carolina State University

Name	Affiliation
Nick Lytle	University of Florida
Pedro Manuel Moreno-Marcos	Universidad Carlos III de Madrid
Ryan Hodson	Durham University
Sa'ar Gershon	Weizmann Institute of Science
Sami Baral	Worcester Polytechnic Institute
Sein Minn	Inria
Shruthi Chockkalingam	University of California, Berkeley
Sonja Ristic	University of Novi Sad
Steven Moore	Carnegie Mellon University
Tahani Aljohani	Durham University
Tasmia Shahriar	North Carolina State University
Warren Li	University of Michigan
Xi Yang	North Carolina State University
Ye Mao	North Carolina State University
Yiqiao Xu	North Carolina State University
Yun Huang	Carnegie Mellon University
Zhi Li	University of California, Berkeley
Zhongtian Sun	Durham University

Sponsors

Platinum

Direction du numérique pour l'éducation, ministère de l'Éducation nationale, de la Jeunesse et des Sports

Gold

Pix

The French National Centre for Scientific Research (CNRS)

National Institute for Research in Digital Science and Technology (Inria)

Silver

Eedi

EvidenceB

Educational Testing Service (ETS)

Duolingo Research

Table of Contents

JEDM Presentations

Full Papers

Short Papers

Posters and Demos

Doctoral Consortium

Industry Papers

Mapping Python Programs to Vectors using Recursive Neural Encodings

Benjamin Paaßen
The University of Sydney
benjamin.paassen@sydney.edu.au

Jessica McBroom
The University of Sydney
jmcb6755@sydney.edu.au

Bryn Jeffries
Grok Learning
bryn@groklearning.com

Irena Koprinska
The University of Sydney
irena.koprinska@sydney.edu.au

Kalina Yacef
The University of Sydney
kalina.yacef@sydney.edu.au

Abstract

Educational data mining involves the application of data mining techniques to student activity. However, in the context of computer programming, many data mining techniques can not be applied because they require vector-shaped input whereas computer programs have the form of syntax trees. In this paper, we present ast2vec, a neural network that maps Python syntax trees to vectors and back, thereby enabling about a hundred data mining techniques that were previously not applicable. Ast2vec has been trained on almost half a million programs of novice programmers and is designed to be applied across learning tasks without re-training, meaning that users can apply it without any need for deep learning. We demonstrate the generality of ast2vec in three settings: First, we provide example analyses using ast2vec on a classroom-sized dataset, involving two novel techniques, namely progress-variance-projection for visualization and a dynamical systems analysis for prediction. In these examples, we also explain how ast2vec can be utilized for educational decisions. Second, we consider the ability of ast2vec to recover the original syntax tree from its vector representation on the training data and two further large-scale programming datasets. Finally, we evaluate the predictive capability of a linear dynamical system on top of ast2vec, obtaining similar results to techniques that work directly on syntax trees while being much faster (constant- instead of linear-time processing). We hope ast2vec can augment the educational data mining toolbelt by making analyses of computer programs easier, richer, and more efficient.

Keywords:

computer science education, computer programs, representation learning, neural networks, visualization, program vectors

Citation

Benjamin Paaßen, Jessica McBroom, Bryn Jeffries, Irena Koprinska, Kalina Yacef. (2021). Mapping Python Programs to Vectors using Recursive Neural Encodings. Journal of Educational Data Mining (JEDM). (to be publisehd).

Benjamin Paaßen, Jessica McBroom, Bryn Jeffries Grok, Irena Koprinska and Kalina Yacef "Mapping Python Programs to Vectors using Recursive Neural Encodings". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 30-30. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Affect, Support and Personal Factors: Multimodal Causal Models of One-on-one Coaching

Lujie Karen Chen
University of Maryland Baltimore County
lujiec@umbc.edu

Joseph Ramsey Carnegie Mellon University
jdramsey@andrew.cmu.edu

Artur Dubrawski Carnegie Mellon University
awd@cs.cmu.edu

Abstract

Human one-on-one coaching involves complex multimodal interactions. Successful coaching requires teachers to closely monitor students' cognitive-affective states and provide support of optimal type, timing, and amount. However, most of the existing human tutoring studies focus primarily on verbal interactions and have yet to incorporate the rich aspects of multimodal cognitive-affective experiences. Meanwhile, the research community lacks principled methods to fully exploit the complex multimodal data to uncover the causal relationships between coaching supports and students' cognitive-affective experiences and their stable individual factors. We explore an analytical framework that is explainable and amenable to incorporating domain knowledge. The proposed framework combines statistical approaches in Sparse Multiple Canonical Correlation, causal discovery and inference methods for observations. We demonstrate this framework using a multimodal one-on-one math problem-solving coaching dataset collected at naturalist home environments involving parents and young children. The insights derived from our analyses may inform the design of effective technology-inspired interventions that are personalized and adaptive

Keywords:

multimodal learning analytics, causal discovery, causal inference, parent coaching, affective and cognitive support

Citation

Lujie Karen Chen, Joseph Ramsey, Artur Dubrawski. (2021). Affect, Support and Personal Factors: Multimodal Causal Models of One-on-one Coaching. Journal of Educational Data Mining (JEDM). (to be publisehd).

Lujie Karen Chen, Joseph Ramsey and Artur Dubrawski "Affect, Support and Personal Factors: Multimodal Causal Models of One-on-one Coaching". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 31-31. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Extending Adaptive Spacing Heuristics to Multi-Skill Items

Benoît Choffin
Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France
benoit.choffin@lisn.upsaclay.fr

Fabrice Popineau
Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France
fabrice.popineau@lisn.upsaclay.fr

Yolaine Bourda
Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France
yolaine.bourda@lisn.upsaclay.fr

Abstract

Adaptive spacing algorithms are powerful tools for helping learners manage their study time efficiently. By personalizing the temporal distribution of retrieval practice of a given piece of knowledge, they improve learners' long-term memory retention compared to fixed review schedules. However, such algorithms are generally designed for the pure memorization of single items, such as vocabulary words. Yet, the spacing effect has been shown to extend to more complex knowledge, such as the practice of mathematical skills. In this article, we extend three adaptive spacing heuristics from the literature for selecting the best skill to review at any timestamp given a student's past study history. In real-world educational settings, items generally involve multiple skills at the same time. Thus, we also propose a multi-skill version for two of these heuristics: instead of selecting one single skill, they select with a greedy procedure the most promising subset of skills to review. To compare these five heuristics, we develop a synthetic experimental framework that simulates student learning and forgetting trajectories with a student model. We run multiple synthetic experiments on large cohorts of 500 simulated students and publicly release the code for these experiments. Our results highlight the strengths and weaknesses of each heuristic in terms of performance, robustness and complexity. Finally, we find evidence that selecting the best subset of skills yields better retention compared to selecting the single best skill to review.

Keywords:

Adaptive spacing, skill selection heuristics, knowledge components, multi-skill learning items

Citation

Benoît Choffin, Fabrice Popineau and Yolaine Bourda. (2021). Extending Adaptive Spacing Heuristics to Multi-Skill Items. Journal of Educational Data Mining (JEDM). (to be publisehd).

Benoît Choffin, Fabrice Popineau and Yolaine Bourda "Extending Adaptive Spacing Heuristics to Multi-Skill Items". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 32-32. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Investigating the Validity of Methods Used to Adjust for Multiple Comparisons in Educational Data Mining

Jeffrey Matayoshi
McGraw Hill ALEKS
Irvine, CA, USA
jeffrey.matayoshi@aleks.com

Shamya Karumbaiah
University of Pennsylvania
Philadelphia, Pennsylvania, USA
shamya@upenn.com

ABSTRACT

Research studies in Educational Data Mining (EDM) often involve several variables related to student learning activities. As such, it may be necessary to run multiple statistical tests simultaneously, thereby leading to the problem of multiple comparisons. The Benjamini-Hochberg (BH) procedure is commonly used in EDM research to address this issue, and it has proven to be a useful method. However, the main limitation of the procedure is that it requires the statistical tests to either be independent or satisfy certain dependency conditions. The Benjamini-Yekutieli (BY) procedure is an alternative that can be applied under arbitrary dependence assumptions, but this extra flexibility comes with a loss of statistical power; hence, the BH procedure is preferred whenever it can be properly applied. Based on these considerations, in this work we employ simulation studies to assess the validity of the BH procedure in two scenarios common to EDM research. The first scenario considers the evaluation and comparison of different classification models—such an analysis might occur, for instance, during the model tuning and validation stage of a study. Then, in the second scenario we look at experiments involving the study of state transitions in sequential data, examples of which occur in affect dynamics research. We find that the BH procedure performs as expected when used with simulated classification model predictions; however, when applied to simulated sequential data, it does not perform at the expected level. Based on these results, as well as previous studies evaluating the BH and BY methods, we discuss the appropriate usage of these procedures for the scenarios under examination.

Keywords

Multiple comparisons, false discovery rate, Benjamini-Hochberg, Benjamini-Yekutieli

1. INTRODUCTION

Consider a statistical analysis that tests several different null hypotheses, either on a single data set, or on related data sets. In such a scenario, the probability of making a *discovery*—i.e., rejecting a null hypothesis—is higher than in an analysis involving a single null hypothesis. Thus, it follows that the probability of rejecting a true null hypothesis increases as well; such errors are variously called *false positives*, *false discoveries*, or *type I errors*. This is known in the statistics literature as the multiple comparisons problem.

Studies in Educational Data Mining (EDM) and related fields are shaping the ongoing research and development of learning systems that are increasingly becoming part of everyday classrooms—thus directly impacting student lives. Greater attention is needed to ensure that the conclusions drawn from these studies are reliable. Along these lines, controlling for multiple comparisons is an important consideration, as it has been argued that addressing the issue is a major factor in ensuring the replicability of scientific results [2]. Additionally, many exaggerated or even incorrect results can be explained by the testing of multiple hypotheses without adjusting for the number of comparisons [34, 40]; while this issue commonly occurs with observational data, experimental studies are not immune to the problem [30].

The main focus of this study is the Benjamini-Hochberg (BH) procedure [3], a method that is commonly applied in EDM research to control the *false discovery rate* (FDR)—defined as the expected rate of false discoveries among all the discoveries made—when multiple statistical tests are used. One complication with using the BH procedure is that, in order for the theoretical guarantees on its performance to hold, the statistical tests must either be independent or satisfy certain dependency conditions [3, 4]. The Benjamini-Yekutieli (BY) procedure is an alternative method that can be used under arbitrary dependence assumptions among the statistical tests [4]. As the BY procedure is more generally applicable than the BH procedure, it is by necessity more conservative and thus less likely to classify a result as being statistically significant; in turn, this causes it to have lower statistical power compared to the BH procedure. Thus, the BH procedure is to be preferred over the BY procedure whenever it can be properly applied.

However, the difficulty is that verifying the conditions for applying the BH procedure is not always straightforward; while some scenarios have been mathematically proven to satisfy these conditions, many common examples have not been. For instance, as of 2010 the case of pairwise comparisons had not been mathematically proven to satisfy the

Jeffrey Matayoshi and Shamya Karumbaiah "Investigating the Validity of Methods Used to Adjust for Multiple Comparisons in Educational Data Mining". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 33-45. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

conditions for using the BH procedure [1], and to the best of our knowledge that has not changed in the interim. Because it's not always clear if the conditions for applying the BH procedure are satisfied, it is often used without any theoretical guarantees on its performance [15]. In other situations, researchers may resort to using both the BH and BY procedures and comparing the results [28]. Motivated by these issues, in this work we investigate two different scenarios that occur within EDM research, with the goal of understanding if the BH procedure is appropriate for each situation. In both scenarios, we assume that a researcher wants to control the FDR, ideally with the BH procedure, but is unsure if it will work as desired. As we are unable to provide mathematical proofs for these scenarios, we instead turn to simulation studies, a procedure that is commonly used to investigate the performance of multiple comparison procedures [1, 3, 14, 22, 31, 32, 38, 39].

The outline of the paper is as follows. We first discuss the specifics of the BH and BY procedures and how to apply them when performing multiple hypothesis tests; additionally, we also look at how multiple comparisons are handled in the EDM community by surveying the literature from the last five EDM conference proceedings. Then, in the remainder of the paper we evaluate the BH and BY procedures for two scenarios that EDM researchers may encounter in their work. The first scenario concerns the usage of these procedures for evaluating and comparing the performance of classification models. In this scenario, we make pairwise comparisons of simulated classifiers, using both accuracy and the area under the receiver operating characteristic curve (AUROC) to evaluate their performance; such a situation can occur, for example, when trying to find the best performing combinations of model algorithms and hyperparameters.

The next scenario we look at is the analysis of state transitions in sequential data. In such an analysis, researchers typically run several hypothesis tests to try and determine the importance of the various transitions between states. Examples of this occur in affect dynamics research, where the BH procedure is commonly used [18, 29]. Here, we run analyses on simulated sequences of transitions using two different statistical measures, and we then apply the BH and BY procedures and compare the results. Finally, based on the results of our numerical experiments, as well as the existing literature on controlling the FDR, we discuss the usage of the BH and BY procedures in these scenarios.

2. CONTROLLING FOR MULTIPLE COMPARISONS

2.1 Benjamini-Hochberg and Benjamini-Yekutieli Procedures

In this study we focus on procedures for controlling the false discovery rate (FDR). The FDR was introduced in [3], and it has since found widespread use in many scientific fields including education research [38], genetics [31, 35], and medical studies [4]. If we let $\mathbf{V}$ be the number of false discoveries and $\mathbf{S}$ be the number of true discoveries, as done in [3] we can define the quantity $\mathbf{Q}$ as

$$\mathbf{Q} = \begin{cases} \frac{\mathbf{V}}{\mathbf{V}+\mathbf{S}}, & \text{if } \mathbf{V}+\mathbf{S} > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Then, the FDR is equal to $\mathbf{E}[\mathbf{Q}]$, the expected proportion of false discoveries among all the discoveries made.

The family-wise error rate (FWER), which is defined as the probability of making at least one false discovery when performing a set of hypothesis tests, is another measure commonly associated with the problem of multiple comparisons. Although the Bonferroni correction is probably the most famous procedure used to control the FWER, there exist many other alternatives. However, while such procedures can be useful in situations in which a false discovery must be avoided at all costs, such as clinical trials of new medical treatments [16], the downside to these methods is a loss of statistical power, resulting in an increased likelihood of missing true discoveries. While procedures for controlling the FWER are concerned with the occurrence of *any* false discoveries, FDR controlling procedures are slightly more permissive, as they allow a certain proportion of the discoveries to be false. Thus, the advantage of FDR controlling procedures is that they typically have greater statistical power and, as such, a better chance of correctly identifying true discoveries; the resulting trade-off is that false discoveries are more likely. However, this trade-off can be beneficial when a large number of hypothesis tests are being conducted,[1] or when the research is of a slightly more exploratory nature.

In addition to introducing the FDR to the scientific literature, the authors in [3] also outlined the BH procedure. As shown there, the BH procedure is mathematically proven to control the FDR at a given level when the statistical tests—or, equivalently, the test statistics—are independent. However, in many practical applications the statistical tests may have some underlying dependence between them. With these situations in mind, further important work on controlling the FDR appeared in [4], where the authors proved that, in addition to the independent case, the BH procedure is valid under certain dependency conditions between the statistical tests. Among other scenarios, it was shown that the BH procedure properly controls the FDR with multivariate normal test statistics having nonnegative correlations. Additionally, the authors in [4] introduced the BY procedure for situations in which the BH procedure is not valid, and they proved that the BY procedure controls the FDR regardless of the dependence between the tests.

In the remainder of this section we discuss the application of the BH and BY procedures. Consider a statistical analysis that involves the testing of m null hypotheses. Of these null hypotheses, $m_0 \leq m$ are true null hypotheses—these correspond to the hypotheses that we expect a statistical test to classify as not being significant—while the remaining $m - m_0$ hypotheses are the false null hypotheses. Note that, in practice, m_0 is an unknown value. Let $P_1, \ldots, P_m$ be the p-values for the m statistical tests, with these values being listed in ascending order; the corresponding null hypotheses are then represented by $H_1, \ldots, H_m$. The relationships

[1] As a relatively extreme example, statistical analyses in genetics research can involve thousands of hypothesis tests, and in such cases FWER controlling procedures can be overly restrictive [1].

between these various terms can be summarized as follows.

	Not significant	Significant	Total
True null	**U**	**V**	m_0
False null	**T**	**S**	$m - m_0$

(2)

- m = total number of hypotheses being tested
- m_0 = number of true null hypotheses
- **V** = number of false positives (i.e., false discoveries or type I errors)
- **S** = number of true positives
- **T** = number of false negatives (i.e., type II errors)
- **U** = number of true negatives

Let q represent our chosen threshold—or, level—for controlling the FDR, and define $\text{FDR}_{\max} = \frac{m_0}{m}q$. If the statistical tests are independent, or if they satisfy certain dependency conditions, it was shown in [4] that the FDR resulting from an application of the BH procedure is at most $\text{FDR}_{\max}$. Such an application works as follows. Assuming once again that the p-values are in ascending order, we find the largest integer k such that $P_k \leq \frac{k}{m}q$. Then, we simply reject all the null hypotheses H_i for which $i \leq k$.

Next, as the BY procedure controls the FDR under arbitrary dependence assumptions, it is necessarily more conservative when rejecting a null hypothesis. This takes the form of a lower threshold for the upper bound used to determine the "significance" of the p-values. Specifically, we find the largest integer k such that $P_k \leq \frac{k}{m \cdot c(m)}q$, where $c(m) = \sum_{i=1}^{m} \frac{1}{i}$. Using this procedure, it was shown in [4] that the resulting FDR is bounded above by $\text{FDR}_{\max} = \frac{m_0}{m}q$, regardless of the type of dependence between the statistical tests.

To see how these procedures work, we next look at an example. Suppose we run 10 separate statistical tests ($m = 10$) that return the following p-values.

$$0.002,\ 0.008,\ 0.011,\ 0.013,\ 0.023,$$
$$0.028,\ 0.092,\ 0.214,\ 0.647,\ 0.853$$

Next, we compare these p-values to the formulas used for the BH and BY thresholds, using a value of $q = 0.1$; for added context, we also include the results for the Bonferroni correction. For each method, the thresholds that correspond to statistically significant p-values are in bold.

k	P_k	BH $\frac{k}{m}q$	BY $\frac{k}{m\sum_{i=1}^{m}\frac{1}{i}}q$	Bonferroni $\frac{1}{m}q$
1	0.002	**0.01**	**0.003**	**0.01**
2	0.008	**0.02**	**0.007**	**0.01**
3	0.011	**0.03**	**0.010**	0.01
4	0.013	**0.04**	**0.014**	0.01
5	0.023	**0.05**	0.017	0.01
6	0.028	**0.06**	0.020	0.01
7	0.092	0.07	0.024	0.01
8	0.214	0.08	0.027	0.01
9	0.647	0.09	0.031	0.01
10	0.853	0.1	0.034	0.01

For the BH procedure, we can see that $k = 6$ is the largest value for which $P_k \leq \frac{k}{m}q$, as we have $0.028 < 0.06$. Thus, the BH procedure, using a value of 0.1, would reject the null hypothesis for the statistical tests corresponding to the lowest six p-values. Next, for the BY procedure we see that $k = 4$ is the largest value for which P_k is less than the corresponding threshold; in this case, we have $0.013 < 0.014$. It's worth noting that, in this example, even though both P_2 and P_3 are *not* below the corresponding thresholds, the BY procedure still classifies them as being statistically significant. This is a feature of FDR controlling procedures that, in many cases, allows them to be more permissive than procedures for controlling the FWER.

2.2 Applications in EDM Research

To understand how EDM research is controlling for multiple comparisons, we reviewed EDM conference proceedings from the last five years (2016–2020). We found that, among the 22 papers that reported controlling for multiple comparisons,[2] half used the Bonferroni correction and half used the BH procedure, with no studies using the BY procedure. Based on the method used to perform the comparisons, the studies can be partitioned as follows: group comparison (8), pairwise comparison (8; including pairwise model comparison), correlation (4), and regression analysis (2). The studies involving group comparisons used statistical methods such as the Mann-Whitney U test, chi-squared test, t-test, and ANOVA. The studies employing pairwise comparisons used methods such as the Kruskal-Wallis test, Mann-Whitney U test, McNemar's test, chi-squared test, and t-test. Overall, these 22 studies investigated diverse educational constructs in virtual learning environments including affect, student behavior in MOOCs, help-seeking, collaboration, and self-regulation.

The choice between the Bonferroni correction and the BH procedure varied in the studies, as the selection was not completely determined by the study methodology. For instance, an exploratory study used the more conservative Bonferroni method for a correlational analysis [61], while an experimental study with group comparisons used the less conservative BH procedure [46]. For EDM research, selecting between the Bonferroni correction and the BH procedure may not be universal and likely depends on the context of the study. As an example, consider that an analysis examining student demographic differences on an important educational construct—such as self-efficacy, affect, or learning—likely has fewer data samples from underrepresented minorities [20]. In such a case, penalizing the statistical power with a more conservative method like the Bonferroni correction may lead to missed opportunities for critical discoveries related to equity. On the other hand, contrast this with the evaluation of an expensive and large-scale educational technology intervention in a public school system; given the costs involved, both financially and otherwise, it could be argued that such an evaluation requires a more conservative approach to control for false discoveries.

More broadly, EDM research may not always involve large data sets. This is particularly true for educational constructs that require resource-intensive data collection procedures—

[2] See Section 8 for the full list of references.

e.g., training coders, gathering approvals, and conducting classroom studies. Hence, using the Bonferroni correction to control for multiple comparisons at the expense of losing statistical power may not always be affordable. In contrast, using the BH procedure in scenarios that violate its statistical assumptions may lead to invalid conclusions. Our review of EDM studies from the last five years also revealed that the field may not be taking advantage of the BY procedure, especially in scenarios where it is difficult to meet the assumptions of the BH procedure. These observations are what motivated us to investigate the use of the BH and BY procedures in research settings relevant to EDM.

3. METHODS

In this section we outline the general procedure we follow for our simulation studies. Since evaluating multiple comparison procedures requires knowledge of whether a null hypothesis is true, and as this isn't typically known with real data, simulations are commonly used for such analyses. In all of our experiments, we begin by generating simulated data according to a given probability distribution. While the specifics of this procedure vary for the two scenarios we consider, the common thread is that this must be done in a way as to have control over whether or not each null hypothesis is true. For example, in our comparisons of simulated classification models, the performance of each model is controlled by a single parameter; thus, when this parameter differs for two models, the null hypothesis that the models perform equally well is false.

Another important detail is that, as we are focusing on two particular scenarios, we can generate simulated data specific to these scenarios. That is, for the model comparison experiments we simulate both the classifier predictions and the ground truth labels; then, for the state transition analysis we generate simulated sequences of states. By simulating the underlying data for each scenario, we are attempting to evaluate the BH and BY procedures in conditions that are as realistic as possible. In comparison, other studies that are more general in nature may simulate the distribution of the test statistics, rather than the underlying data, when evaluating multiple comparison procedures.

After generating the data for a simulation run, we perform our statistical tests and compute the corresponding p-values. Once this is done, we then apply the BH and BY procedures for various threshold values q—specifically, we use 0.05, 0.1, and 0.15 in all our evaluations. While a value of 0.05 is commonly used, it's been argued that this threshold may be too low for some applications [26]; thus, we evaluate a range of values in our simulations. Based on the statistical significance results from our application of the BH and BY procedures, we can compute $\mathbf{Q}$, the proportion of false discoveries among all the discoveries made, using (1). To obtain our estimate of the FDR, we then compute the average of $\mathbf{Q}$ over a total of 10,000 simulation runs. For the various values of q, we compare these FDR estimates to the values of FDR_{max} as defined in Section 2.1.

At this point, it's worth mentioning that the value of $\mathbf{Q}$—and, hence, the estimated FDR value—can be very different from the false positive rate.[3] Using the notation in (2), the false positive rate can be written as $\frac{\mathbf{V}}{\mathbf{V}+\mathbf{U}}$. In comparison, $\mathbf{Q}$ is computed with the formula $\frac{\mathbf{V}}{\mathbf{V}+\mathbf{S}}$, which has a different denominator. Thus, while the FDR is the expected proportion of false discoveries among all the rejected null hypotheses, the false positive rate is the (expected) proportion of false discoveries among all the true null hypotheses. Consider the following example. Assume we are testing 20 total hypotheses, all of which are true null hypotheses ($m_0 = m = 20$). Furthermore, assume that one false positive is recorded. Then, the false positive rate for this set of tests would be equal to $\frac{1}{1+19} = 0.05$. However, applying (1) gives a value of $\mathbf{Q} = \frac{1}{1+0} = 1$. This discrepancy is something to keep in mind as we analyze the results from our simulation studies in subsequent sections.

4. MODEL COMPARISONS

The first scenario we study concerns the comparison of several classification models on a fixed set of test or validation data. A common example of this occurs during the model building process, where it may be necessary to evaluate the performance of many different combinations of classification models and hyperparameters. In such a case, it can be helpful for the researcher to run statistical tests to more precisely quantify the differences in performance. To that end, we focus on the pairwise comparisons of the classifiers, where we assume that the classifiers could have different underlying algorithms—e.g., logistic regression vs. random forest—or the same algorithm with different hyperparameters. We evaluate each pair of classifiers by looking at both the accuracy and the area under the receiver operating characteristic curve (AUROC). To measure the possible difference between the accuracy values of the models, we use McNemar's test [13, 27]. When conducting pairwise comparisons of classifier accuracy on a fixed set of test data—as opposed to a procedure such as k-fold cross-validation, where the test set varies—using McNemar's test is recommended [10]; for these evaluations we use the implementation in the `statsmodels` [33] Python library. Then, to compare the AUROC values we use DeLong's test [9], a method developed to statistically test for differences in AUROC values; specifically, we apply the fast version of the algorithm outlined in [36].[4]

Our simulations use the following procedure. We assume that we are evaluating the performance of a binary classifier on a test set containing n data points; for these simulations we use n-values of 500, 1000, and 5000. For each value of n, we sample n numbers uniformly at random from 0.01 to 0.99; we refer to this set of numbers as $\mathbf{U}_n$. In each simulation run, the numbers in $\mathbf{U}_n$ are used to generate the labels for our data using the following procedure. Let i be an integer from 1 to n, and let $p_i \in \mathbf{U}_n$. With probability p_i we assign a label of 1 to y_i; otherwise, with probability $1-p_i$ it is then given a label of 0. Note that the set $\mathbf{U}_n$ is generated once for each value of n, and this same set is then used repeatedly for all of our simulation runs with a test set of size n.

[3] That is, while "false discovery" and "false positive" are used interchangeably, the terms "false discovery rate" and "false positive rate" have different definitions.

[4] The code for our implementation of the algorithm in [36], as well as for running all of our experiments, is available at https://github.com/jmatayoshi/multiple-comparisons.

Table 1: Accuracy and AUROC values for an example simulation run using a test set of size $n = 1000$.

σ	0.1	0.1	0.1	0.5	1	2
Accuracy	0.733	0.724	0.732	0.706	0.651	0.606
AUROC	0.824	0.821	0.824	0.787	0.721	0.655

We next describe our procedure for simulating the classifier predictions. Let c_{ij} represent the predicted probability given by classifier j for the i-th data point in our test set. To generate c_{ij}, we begin by converting $p_i \in \mathbf{U}_n$ to a $\mathbf{z}$-score. Then, to add noise to the classifier's prediction we randomly sample a value, s_{ij}, from a normal distribution with mean 0 and standard deviation σ_j, add this to the $\mathbf{z}$-score, and then convert everything back to a probability; the resulting value is c_{ij}. The size of σ_j controls the performance of the classifier, with lower values giving predicted probabilities that are less noisy and more likely to align with the class labels. Let Φ denote the cumulative distribution function (CDF) of the standard normal distribution. Our procedure for generating the classifier predictions can be summarized as follows.

1. $z_i = \Phi^{-1}(p_i)$
2. Draw sample value s_{ij} from $\mathcal{N}(0, \sigma_j^2)$
3. $c_{ij} = \Phi(z_i + s_{ij})$

To get an idea of the effect of different values of σ on the performance of our simulated classifier predictions, in Table 1 we show the accuracy and AUROC values from one simulation run, using different values of σ and a test set size of $n = 1000$. The three classifiers with σ values of 0.1 have the best performance, with accuracy values from 0.72 to 0.73 and AUROC values around 0.82. The other classifiers, to varying degrees, perform worse, with the lowest accuracy and AUROC values at roughly 0.61 and 0.66, respectively. Our initial analysis simulates the pairwise comparison of six different classification models, where all the classifiers are assumed to perform equally; specifically, we use a value of $\sigma = 0.5$ for each model. Using our previously described procedure, we generate a total of 10,000 simulation runs for each value of n. Our experimental setup results in $\binom{6}{2} = 15$ pairwise comparisons ($m = 15$), and as there are no underlying differences between the simulated classifiers, we have 15 true null hypotheses ($m_0 = 15$). As such, if the conditions for the BH procedure are satisfied, we expect the FDR to be less than $\text{FDR}_{\text{max}} = \frac{15}{15}q = q$. The results are shown in Figures 1 and 2, where we display the estimated FDR rates for the BH and BY procedures, for different combinations of test set sizes and values of q. Using both McNemar's test and DeLong's test, the BH procedure appears to control the FDR by keeping it below the corresponding FDR_{max} value, shown by the dashed line, in all cases—that is, for all combinations of test set sizes and q. In comparison, the BY procedure is much more conservative, with each estimated FDR value far below the FDR_{max} line.

For our second set of simulations, we use the values of σ that appear in Table 1 to generate six different models. As there are three models with the same value of $\sigma = 0.1$, we have $\binom{3}{2} = 3$ true null hypotheses ($m_0 = 3$) out of 15 total comparisons ($m = 15$). Thus, under the appropriate conditions the BH procedure should keep the FDR at or below $\text{FDR}_{\text{max}} = \frac{3}{15}q = \frac{1}{5}q$. The results are given in Figures 3 and 4, where we can see that the estimated FDR values using the BH procedure are at or below the value of FDR_{max}, given by the dashed line, in all cases—that is, for all combinations of test set sizes and q. As before, the estimated FDR values from the BY procedure are very low, with each value again appearing far below the corresponding FDR_{max} line.

These results are seemingly consistent with previous works analyzing the performance of the BH procedure with pairwise comparisons [21, 38]. The findings from several of these studies are summarized in [39], where the author states that in "all the studies, for all configurations of true and false hypotheses simulated, for balanced and for non-balanced designs, normal and non-normal distributions, the BH procedure controlled the FDR." Thus, combining these previous results with our experiments from this section, there appears to be good evidence that the BH procedure properly controls the FDR in the case of pairwise comparisons of classification models. We return to this subject in the discussion.

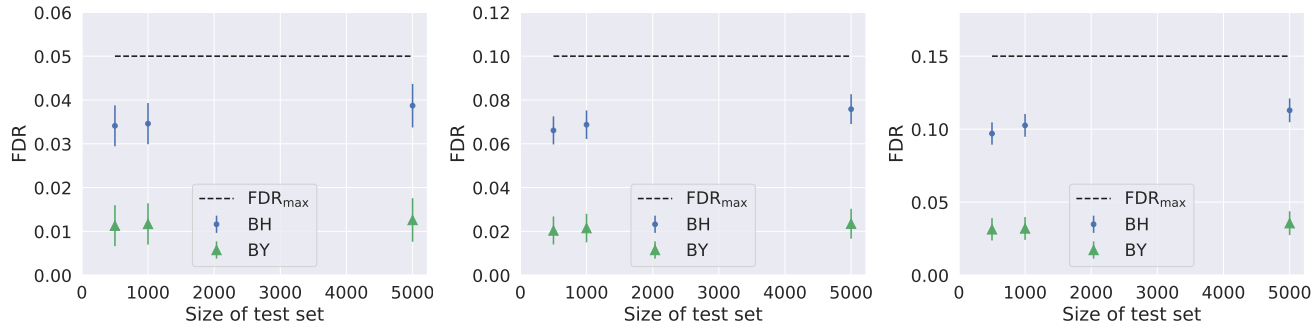
5. TRANSITIONS IN SEQUENTIAL DATA

In our second scenario we look at data that are sequential in nature, as examples of such data appear in many areas of educational research. One particular focus with sequential data is the analysis of transitions between different states—or events—in these sequences. Researchers are often interested in understanding if transitions between certain pairs of states are significant, either because they happen often or because they rarely appear. Typically in such cases, many pairs of states are evaluated with statistical tests, thus necessitating a correction for multiple comparisons. For example, past studies have analyzed logs of student actions in learning systems, in an attempt to understand the differences between productive and unproductive transitions between activities within these systems [5, 6]. Another example is affect dynamics research, which studies sequences of student affective states, with the goal of understanding how students transition between these different states. Previous works in this area have used the BH procedure to control the FDR [18, 29], and as such the goal of our next analysis is to investigate the appropriateness of using this procedure when analyzing state transitions.

5.1 Experimental Setup

Our numerical experiments for sequential data evaluate the BH and BY procedures on simulated sequences of states. Each of these sequences could represent, for example, a student's affective states while working in a learning system. The states are randomly sampled according to the probability distribution given in Table 2; each entry in the table gives the probability of sampling the next state (column) based on the value of the previous state (row). For example, suppose that C is the previous state. In this case, A has a probability of 0.2 of being the next state, B has a probability of $0.2 - \gamma$ of being the next state, and so on.

For our simulations, we use two different values for γ: 0, which results in all 25 hypotheses being true null hypotheses; and 0.05, which results in 21 true null hypotheses, out of the 25. For each value of γ, we generate n sequences consisting of 20 states each. To generate these sequences, the first state in each sequence is sampled randomly from the five

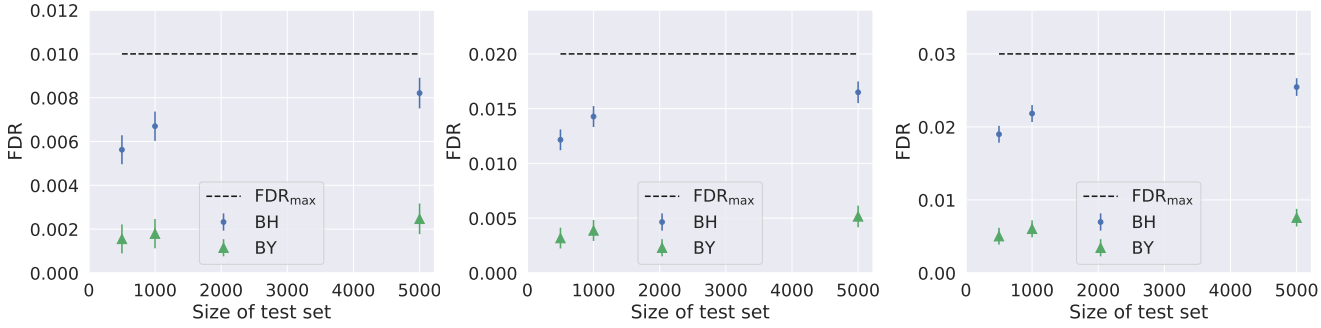


(a) $q = 0.05$ (b) $q = 0.1$ (c) $q = 0.15$

Figure 1: Comparison of the estimated FDR for the BH and BY procedures, using McNemar's test and six classifiers with the same value of $\sigma = 0.5$. Vertical lines represent the 99% confidence interval for each estimated FDR value.

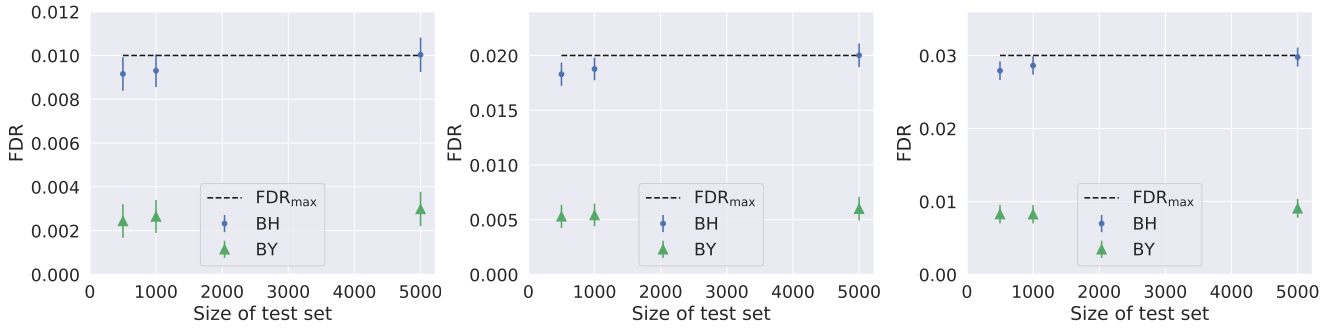
(a) $q = 0.05$ (b) $q = 0.1$ (c) $q = 0.15$

Figure 2: Comparison of the estimated FDR for the BH and BY procedures, using DeLong's test and six classifiers with the same value of $\sigma = 0.5$. Vertical lines represent the 99% confidence interval for each estimated FDR value.



(a) $q = 0.05$ (b) $q = 0.1$ (c) $q = 0.15$

Figure 3: Comparison of the estimated FDR for the BH and BY procedures, using McNemar's test and the σ values in Table 1. Vertical lines represent the 99% confidence interval for each estimated FDR value.



(a) $q = 0.05$ (b) $q = 0.1$ (c) $q = 0.15$

Figure 4: Comparison of the estimated FDR for the BH and BY procedures, using DeLong's test and the σ values in Table 1. Vertical lines represent the 99% confidence interval for each estimated FDR value.

Table 2: Probability distribution used to generate the simulated sequences of states. Each entry represents the probability of making a transition to the next state (column), given the previous state (row).

prev \ next	A	B	C	D	E
A	0.2	$0.2+\gamma$	0.2	$0.2-\gamma$	0.2
B	0.2	0.2	0.2	0.2	0.2
C	0.2	$0.2-\gamma$	0.2	$0.2+\gamma$	0.2
D	0.2	0.2	0.2	0.2	0.2
E	0.2	0.2	0.2	0.2	0.2

Table 3: Marginal model coefficient p-values from one simulation run using $\gamma = 0.05$. With a threshold of $q = 0.05$, both the BH and BY procedures give the same statistical significance results for this example; namely, only the four transition pairs with sample probabilities modified by γ are statistically significant.

prev \ next	A	B	C	D	E
A	0.252	0.000	0.335	0.000	0.703
B	0.496	0.365	0.327	0.864	0.252
C	0.035	0.000	0.527	0.000	0.569
D	0.260	0.652	0.080	0.980	0.889
E	0.581	0.099	0.800	0.869	0.179

choices, and then all subsequent states are sampled according to the probability distribution in Table 2. For each set of n sequences we evaluate our statistical tests (described in Sections 5.2 and 5.3) and then compute the resulting value for $\mathbf{Q}$; this constitutes one simulation run. We then perform 10,000 simulation runs for each value of n in order to obtain an estimate of the true FDR. For this analysis, we use the following values of n: 50, 100, and 200.

The L statistic, originally introduced in [12], is intended to be used as a measure of the significance of different pairs of transitions, and it has been widely applied in the study of affect dynamics [11, 12, 18]. Given two states A and B, it measures the likelihood of transitions from A to B while taking into account the overall frequency at which B occurs. However, several recent works have revealed issues with the use of the L statistic for the analysis of state transitions [7, 18, 19]. Thus, for our simulations we use two newer methods that have been developed in response to the problems with the L statistic. First, in Section 5.2 we look at the performance of the BH procedure when used in combination with the marginal model approach outlined in [25]. Then, in Section 5.3 we evaluate the BH procedure when it is used with the modified version of the L statistic from [24].

5.2 Marginal Model

To estimate the influence that starting in state A has on the probability of making a transition to B, in this section we use the marginal model regression procedure from [25]. In this approach, the regression model has a binary response variable, where the value of this variable is one if the next state is equal to B, and it is zero otherwise. Based on the binary response variable, we use the logit as our link function. Our predictor—or, independent—variable is also binary, with a value of one if the previous state is equal to A and zero otherwise. We can summarize this procedure as follows.

- $y = y_{it}$: one if B is the next state for student i at time t; zero otherwise
- $x = x_{it}$: one if A is the previous state for student i at time t; zero otherwise

Letting S represent the standard logistic function, the regression equation then has the form

$$P(y_{it} = 1 \,|\, x_{it}) = S(\beta_0 + \beta_1 x_{it}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{it})}}. \quad (3)$$

When $x_{it} = 1$ the regression model returns an estimate for $P(B \,|\, A)$, the probability of a transition to B, given that the starting state is A. Then, when $x_{it} = 0$ it returns an estimate for $P(B \,|\, \overline{A})$, the probability of a transition to B, given that the starting state is not A. Thus, to measure the importance of starting in state A, we focus on testing if the value of β_1 is significantly different from zero. This is done using a two-tailed z-test on the value of β_1 for each individual fit of the regression model.

Finally, as the sequential data used in these analyses typically take the form of repeated measurements on a student,

the result is a set of dependent—or correlated—data. To account for this dependence, as outlined in [25] we use a marginal model, based on generalized estimating equations (GEE) [17, 23], to estimate the logistic regression coefficients; in particular, we use the GEE implementation from the `statsmodels` Python library.

As before, let m denote the total number of statistical tests, with $m_0 \leq m$ representing the number of true null hypotheses. Using the BH procedure with a value of $\gamma = 0$, we have $m_0 = m$; as such, we would expect the FDR to be less than $\text{FDR}_{\max} = \frac{25}{25}q = q$ if the BH conditions are satisfied. Then, for all values of $\gamma > 0$ we would expect the FDR to be less than $\text{FDR}_{\max} = \frac{21}{25}q$, assuming the BH conditions are satisfied, as $m_0 = 21$ of the tests are true null hypotheses.

The first set of results, using a value of $\gamma = 0$, is shown in Figure 5. Here, we can see that in all cases the estimated FDR values from the BH procedure are above the theoretical upper bound of $\text{FDR}_{\max}$, shown by the dashed line. The gap is particularly notable with smaller numbers of sequences. On the other hand, the BY procedure offers much more stringent control of the FDR, with all of the estimated values appearing below the $\text{FDR}_{\max}$ line. Figure 6 then shows the results from using a value of $\gamma = 0.05$. Overall, the picture appears similar to the $\gamma = 0$ case, with the estimated FDR values from the BH procedure always appearing above the $\text{FDR}_{\max}$ line, and with the difference again being more pronounced with smaller numbers of sequences.

5.3 Removing Self-Transitions

Our final set of experiments investigates a specific situation in sequential data analysis that occurs when researchers want to remove the influence of repeated states. To do this, many researchers in the affect dynamics community remove *self-transitions*—i.e., transitions where the same state is repeated for more than one step—before analyzing the data [18]. However, this procedure has been shown to overestimate the significance of transitions when used with the L statistic [19]. Thus, for this analysis we instead use a modified version of the L statistic, named L^* [24].

DEFINITION 1. *Let A and B be two states, and let*

$$T_{\overline{A}} = \{\textit{transitions where the next state is not } A\}. \tag{4}$$

Then, we define

$$L^*(A \to B) := \frac{P(B \mid A, T_{\overline{A}}) - P(B \mid T_{\overline{A}})}{1 - P(B \mid T_{\overline{A}})}, \tag{5}$$

where $P(B \mid A, T_{\overline{A}})$ is the probability of a transition to B in $T_{\overline{A}}$, given that the starting state is A, while $P(B \mid T_{\overline{A}})$ is the overall probability of a transition to B in $T_{\overline{A}}$.

The base rate of the state B, given by $P(B \mid T_{\overline{A}})$ in (5), can be computed either individually for each sequence, or averaged over the entire set of sequences. For the computations in the remainder of this work, we compute these rates individually per sequence.

Our analysis using L^* applies the statistic to the sequences from our experiments in Section 5.2. Specifically, we take each sequence and, for each pair of transition states, compute (5). To test for statistical significance, we follow the procedure outlined in [24] and apply a two-tailed t-test to the L^* values. The results for the $\gamma = 0$ and $\gamma = 0.05$ sequences are shown in Figures 7 and 8, respectively. While perhaps not quite as prominent as with the marginal model procedure, there are several examples where the estimated FDR values from the BH procedure are clearly above the $\text{FDR}_{\max}$ line. As with the marginal model procedure, the worst cases occur with the smallest number of sequences.

5.4 Dependence of the Statistical Tests

The experiments in this section provide evidence that, when used in combination with either the marginal model procedure or L^*, the BH procedure does not always control the FDR at the desired level; in turn, this may indicate that the conditions for applying the BH procedure are not satisfied. In the remainder of this section, we outline two arguments that show the assumption of independence is violated between the statistical tests used in these analyses. Note that these are not rigorous mathematical proofs; rather, our goal here is to simply give some intuition into the relationships between the statistical tests.

Consider a set of sequential data consisting of possible states A, B, C, D, and E. For states A and B, let $\beta_{A,B}$ represent the value of β_1 in (3) for transitions of the form $A \to B$. Suppose that the following inequalities hold.

$$\begin{aligned} \beta_{A,A} &> 0 & \beta_{A,B} &> 0 \\ \beta_{A,C} &> 0 & \beta_{A,D} &> 0 \end{aligned} \tag{6}$$

Consider, for example, $\beta_{A,B}$. The corresponding marginal model estimates the probability of a transition to B, depending on whether or not the starting state is A—these estimates correspond to $P(B \mid A)$ and $P(B \mid \overline{A})$, respectively. The inequalities in (6) can then be interpreted as follows.

$$\begin{aligned} P(A \mid A) &> P(A \mid \overline{A}) & P(B \mid A) &> P(B \mid \overline{A}) \\ P(C \mid A) &> P(C \mid \overline{A}) & P(D \mid A) &> P(D \mid \overline{A}) \end{aligned} \tag{7}$$

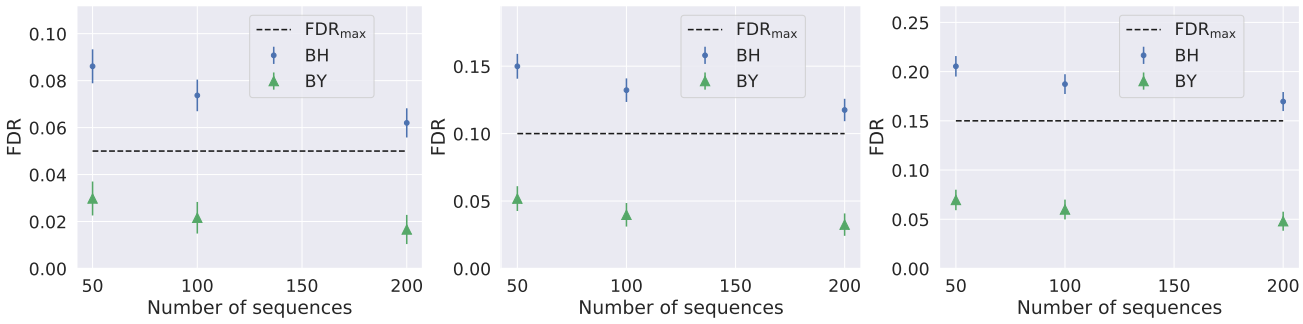
Next, consider the following two equalities.

$$\begin{aligned} P(E \mid A) &= 1 - P(A \mid A) - P(B \mid A) - P(C \mid A) - P(D \mid A) \\ P(E \mid \overline{A}) &= 1 - P(A \mid \overline{A}) - P(B \mid \overline{A}) - P(C \mid \overline{A}) - P(D \mid \overline{A}) \end{aligned} \tag{8}$$

Combining (7) and (8), it follows that $P(E \mid A) < P(E \mid \overline{A})$, or, equivalently, that $\beta_{A,E} < 0$. What this argument illustrates is that it's not possible—or, at least, it's highly unlikely—for $\beta_{A,E}$ to be positive when the other four coefficients are positive, which means that the corresponding statistical tests are not completely independent of each other.

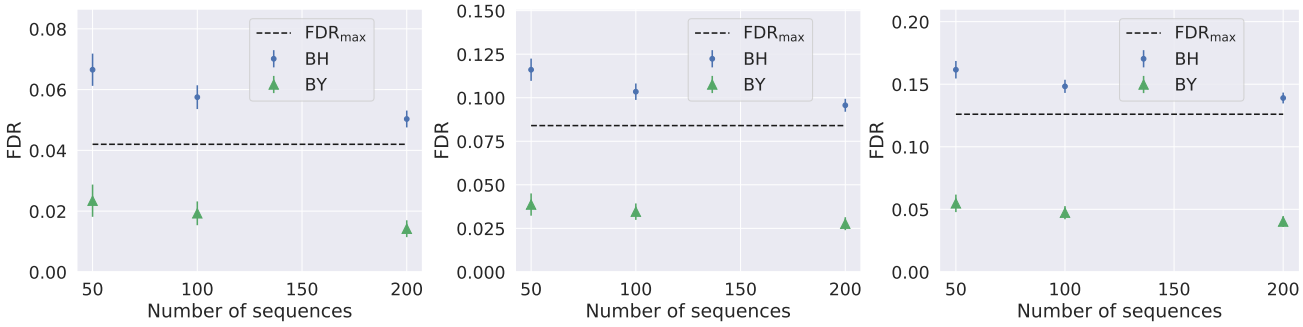
Next, suppose we are in the situation of removing self-transitions and applying L^*; thus, in what follows assume we are interested in transitions from A to B and that, following (4) in Definition 1, all transitions to A have been removed from our sequence. Suppose the following inequalities hold.

$$\begin{aligned} P(B \mid A) &> P(B) \\ P(C \mid A) &> P(C) \\ P(D \mid A) &> P(D) \end{aligned} \tag{9}$$



(a) $q = 0.05$ (b) $q = 0.1$ (c) $q = 0.15$

Figure 5: Comparison of the estimated FDR for the BH and BY procedures, using a value of $\gamma = 0$ and the marginal model method. Vertical lines represent the 99% confidence interval for each estimated FDR value.



(a) $q = 0.05$ (b) $q = 0.1$ (c) $q = 0.15$

Figure 6: Comparison of the estimated FDR for the BH and BY procedures, using a value of $\gamma = 0.05$ and the marginal model method. Vertical lines represent the 99% confidence interval for each estimated FDR value.

Consider the equalities

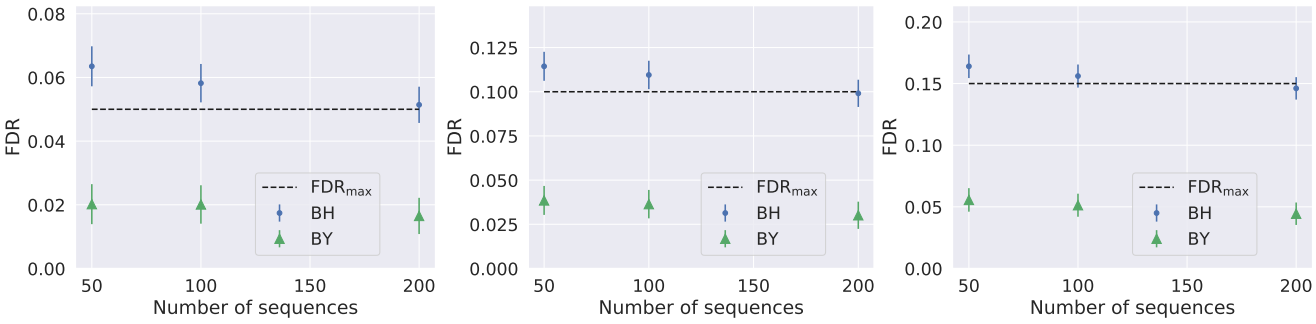
$$\begin{aligned} P(E \mid A) &= 1 - P(B \mid A) - P(C \mid A) - P(D \mid A) \\ P(E) &= 1 - P(B) - P(C) - P(D), \end{aligned} \tag{10}$$

where we're using the fact that, as we are removing transitions to A, $P(A \mid A) = 0$ and $P(A) = 0$. Combining (9) and (10), it follows that $P(E \mid A) < P(E)$. Thus, it's not possible for all four of the conditional probabilities to be larger than the base probabilities—in turn, this means that at least one of the L^* values must be negative. As such, it follows that the corresponding statistical tests are not completely independent of each other.

6. DISCUSSION

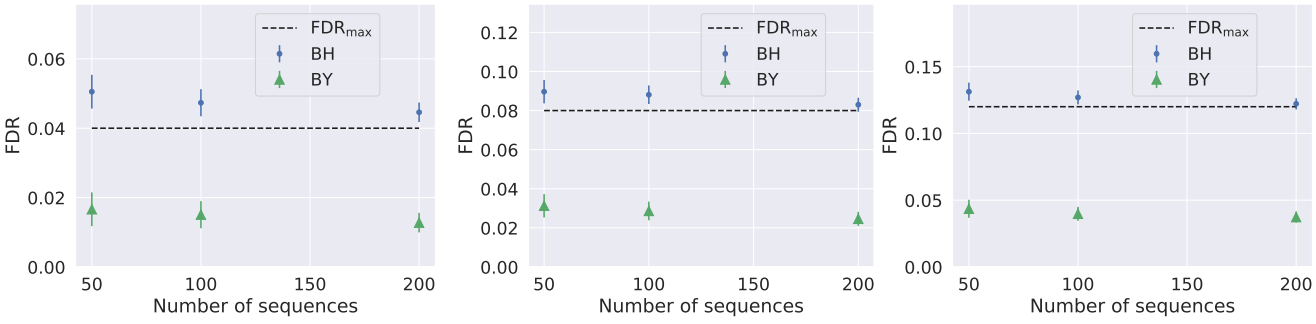
In this paper, we investigated the validity of methods used to adjust for false discoveries when performing multiple comparisons. In two scenarios relevant to EDM research, we evaluated the performance of the commonly used BH procedure in relation to an alternate method—the BY procedure—that is more general and is valid to use when the assumptions of the BH procedure cannot be met. Our first set of experiments looked at the performance of these procedures when used with pairwise comparisons of classification models on a fixed set of test data. In all our experiments, using both accuracy and AUROC as our performance metrics, the BH procedure controlled the FDR at the expected level. These results are consistent with previous studies investigating pairwise comparisons, where in all cases the BH procedure properly controlled the FDR [21, 38, 39]. Combining these previous results with the experiments in this study, our current view is that the usage of the BH procedures appears justified in this scenario—that is, one can reasonably expect the BH procedure to properly control the FDR when performing pairwise comparisons of classifiers on a fixed set of test data.

Contrast this with our investigation on sequential data, where we observed that the BH procedure, when combined with either the marginal model procedure or L^*, did not control the FDR at the expected level—this happened with various experimental conditions and for various threshold values q. The results could be an indication that the theoretical conditions for applying the BH procedure might not be satisfied in these situations. Combined with the fact that various issues involving the analysis of state transitions have recently come to light [7, 18, 19, 24, 25], we believe that using the more conservative BY procedure is justified, particularly when the analysis involves a small number of sequences. To compensate for the fact that it is more conservative, when applying the BY procedure we suggest the use of a larger value of q, such as 0.1.



(a) $q = 0.05$ (b) $q = 0.1$ (c) $q = 0.15$

Figure 7: Comparison of the estimated FDR for the BH and BY procedures, using a value of $\gamma = 0$ and the L^* statistic. Vertical lines represent the 99% confidence interval for each estimated FDR value.



(a) $q = 0.05$ (b) $q = 0.1$ (c) $q = 0.15$

Figure 8: Comparison of the estimated FDR for the BH and BY procedures, using a value of $\gamma = 0.05$ and the L^* statistic. Vertical lines represent the 99% confidence interval for each estimated FDR value.

More generally, it's worth noting that there are many examples where the BH procedure performs well without any theoretical guarantees [14, 22]. Thus, for situations in which the BH procedure has not been theoretically or empirically vetted, we offer a couple of suggestions. First, whenever possible, conducting a simulation study may be helpful; as seen in this work, the results could give evidence for or against the usage of the BH procedure. Failing that, and if there is good reason to doubt the validity of using the BH procedure, we suggest that the BY procedure be considered as a possible alternative. In these cases, a higher value for q may be justified in order to compensate for the more restrictive nature of the BY procedure, and this decision could be made based on the context of the study. For instance, in studies that are exploratory in nature or have small sample sizes, the loss of statistical power might be a larger concern; thus, the BY procedure using a threshold of 0.1 or larger may be appropriate. Whereas, in an experimental study looking for conclusive evidence, it may be preferable to use the BY procedure with a smaller value of q.

In regards to future work in this area, it would be of interest to more completely understand why the BH procedure fails to properly control the FDR in our simulations with sequential data. While we presented an argument in Section 5.4 that showed the statistical tests are not independent, it's an open question whether this argument can be extended to rigorously show that the assumptions of the BH procedure are violated—we are currently looking at this in more detail. Furthermore, it's possible that other elements may also be at play. For example, as discussed previously there are known issues with several existing methods commonly used to evaluate state transitions. While the methods we used in this study were originally developed in response to these problems [24, 25], it's possible that these existing issues, or perhaps even new ones, are a factor; thus, further adjustments to the marginal model and L^* methods could lead to improved control of the FDR with the BH procedure.

There exist other directions for future work that we are currently exploring. First, as the literature on multiple comparisons and controlling the FDR is actively growing, many methods have been developed over the years. Thus, while the BH and BY procedures are arguably the most notable of the FDR controlling procedures, it would be worthwhile to evaluate some of the newer alternatives, especially for the analysis of state transitions. Second, our analyses in this work focused exclusively on false discoveries (Type I errors) and did not consider false negatives (Type II errors). As such, in future work we aim to explicitly examine the interaction between these two types of errors with respect to the BH and BY procedures and EDM research.

7. REFERENCES

[1] Y. Benjamini. Discovering the false discovery rate. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):405–416, 2010.

[2] Y. Benjamini. Selective inference: The silent killer of replicability. *Harvard Data Science Review*, 2(4), 12 2020. https://hdsr.mitpress.mit.edu/pub/l39rpgyc.

[3] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995.

[4] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, pages 1165–1188, 2001.

[5] G. Biswas, H. Jeong, J. Kinnebrew, B. Sulcer, and R. D. Roscoe. Measuring self-regulated learning skills through social interactions in a teachable agent environment. *Res. Pract. Technol. Enhanc. Learn.*, 5:123–152, 2010.

[6] N. Bosch and S. D'Mello. The affective experience of novice computer programmers. *International Journal of Artificial Intelligence in Education*, 27(1):181–206, 2017.

[7] N. Bosch and L. Paquette. What's next? Edge cases in measuring transitions between sequential states. 2020. Submitted for publication.

[8] C. Cody, M. Maniktala, D. Warren, M. Chi, and T. Barnes. Does autonomy help Help? The impact of unsolicited hints and choice on help avoidance and learning. In *Proceedings of the 13th International Conference on Educational Data Mining*, pages 591–595, 2020.

[9] E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, pages 837–845, 1988.

[10] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[11] S. D'Mello and A. Graesser. Dynamics of affective states during complex learning. *Learning and Instruction*, 22(2):145–157, 2012.

[12] S. D'Mello, R. S. Taylor, and A. Graesser. Monitoring affective trajectories during complex learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society, 29 (29)*, pages 203–208, 2007.

[13] A. L. Edwards. Note on the correction for continuity in testing the significance of the difference between correlated proportions. *Psychometrika*, 13(3):185–187, 1948.

[14] A. Farcomeni. More powerful control of the false discovery rate under dependence. *Statistical Methods and Applications*, 15(1):43–73, 2006.

[15] W. Fithian and L. Lei. Conditional calibration for false discovery rate control under dependence. *arXiv preprint arXiv:2007.10438*, 2020.

[16] J. J. Goeman and A. Solari. Multiple hypothesis testing in genomics. *Statistics in medicine*, 33(11):1946–1978, 2014.

[17] P. J. Heagerty and S. L. Zeger. Marginalized multilevel models and likelihood inference (with comments and a rejoinder by the authors). *Statistical Science*, 15(1):1–26, 2000.

[18] S. Karumbaiah, J. Andres, A. F. Botelho, R. S. Baker, and J. S. Ocumpaugh. The implications of a subtle difference in the calculation of affect dynamics. In *Proceedings of the 26th International Conference on Computers in Education*, pages 29–38, 2018.

[19] S. Karumbaiah, R. S. Baker, and J. Ocumpaugh. The case of self-transitions in affective dynamics. In *Artificial Intelligence in Education-20th International Conference, AIED 2019*, pages 172–181, 2019.

[20] S. Karumbaiah, J. Ocumpaugh, and R. S. Baker. The influence of school demographics on the relationship between students help-seeking behavior and performance and motivational measures. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 99–108, 2019.

[21] H. Keselman, R. Cribbie, and B. Holland. The pairwise multiple comparison multiplicity problem: An alternative approach to familywise and comparison wise Type I error control. *Psychological Methods*, 4(1):58, 1999.

[22] K. I. Kim and M. A. van de Wiel. Effects of dependence in high-dimensional multiple testing problems. *BMC bioinformatics*, 9(1):1–12, 2008.

[23] K.-Y. Liang and S. L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22, 1986.

[24] J. Matayoshi and S. Karumbaiah. Adjusting the L statistic when self-transitions are excluded in affect dynamics. *Journal of Educational Data Mining*, 12(4):1–23, Dec. 2020.

[25] J. Matayoshi and S. Karumbaiah. Using marginal models to adjust for statistical bias in the analysis of state transitions. In *LAK21: 11th International Learning Analytics and Knowledge Conference*, pages 449– 455, 2021.

[26] J. McDonald. *Handbook of Biological Statistics (3rd ed.)*. Sparky House Publishing, 2014.

[27] Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

[28] G. Nolte, O. Bai, L. Wheaton, Z. Mari, S. Vorbach, and M. Hallett. Identifying true brain interaction from eeg data using the imaginary part of coherency. *Clinical neurophysiology*, 115(10):2292–2307, 2004.

[29] J. Ocumpaugh, J. M. Andres, R. Baker, J. DeFalco, L. Paquette, J. Rowe, B. Mott, J. Lester, V. Georgoulas, K. Brawner, et al. Affect dynamics in military trainees using vMedic: From engaged concentration to boredom to confusion. In *International Conference on Artificial Intelligence in Education*, pages 238–249. Springer, 2017.

[30] S. J. Pocock, M. D. Hughes, and R. J. Lee. Statistical problems in the reporting of clinical trials. *New England Journal of Medicine*, 317(7):426–432, 1987.

[31] A. Reiner-Benaim. FDR control by the BH procedure for two-sided correlated tests with implications to gene expression data analysis. *Biometrical Journal*, 49(1):107–126, 2007.

[32] A. Reiner-Benaim, D. Yekutieli, N. E. Letwin, G. I. Elmer, N. H. Lee, N. Kafkafi, and Y. Benjamini. Associating quantitative behavioral traits with gene expression in the brain: Searching for diamonds in the hay. *Bioinformatics*, 23(17):2239–2246, 2007.

[33] S. Seabold and J. Perktold. Statsmodels: Econometric and statistical modeling with Python. In *9th Python in Science Conference*, 2010.

[34] G. D. Smith and S. Ebrahim. Data dredging, bias, or confounding: They can all get you into the BMJ and the Friday papers, 2002.

[35] J. D. Storey and R. Tibshirani. Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences*, 100(16):9440–9445, 2003.

[36] X. Sun and W. Xu. Fast implementation of DeLong's algorithm for comparing the areas under correlated receiver operating characteristic curves. *IEEE Signal Processing Letters*, 21(11):1389–1393, 2014.

[37] R. Venant and M. d'Aquin. Towards the prediction of semantic complexity based on concept graphs. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 188–197, 2019.

[38] V. S. Williams, L. V. Jones, and J. W. Tukey. Controlling error in multiple comparisons, with examples from state-to-state differences in educational achievement. *Journal of educational and behavioral statistics*, 24(1):42–69, 1999.

[39] D. Yekutieli. False discovery rate control for non-positively regression dependent test statistics. *Journal of Statistical Planning and Inference*, 138(2):405–415, 2008.

[40] S. S. Young and A. Karr. Deming, data and observational studies: A process out of control and needing fixing. *Significance*, 8(3):116–120, 2011.

8. EDM REVIEW REFERENCES

[41] H. Anderson, A. Boodhwani, and R. Baker. Assessing the fairness of graduation predictions. In *Proceedings of the 12th International Conference on Educational Data Mining*, 2019.

[42] J. Andrews-Todd, C. Forsyth, J. Steinberg, and A. Rupp. Identifying profiles of collaborative problem solvers in an online electronics environment. In *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[43] A. Bauer, J. Flatten, and Z. Popovic. Analysis of problem-solving behavior in open-ended scientific-discovery game challenges. In *Proceedings of the 10th International Conference on Educational Data Mining*, 2017.

[44] N. Bosch, W. Crues, and N. Shaik. Diverse learners, diverse motivations: Exploring the sentiment of learning objectives. In *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[45] A. F. Botelho, R. Baker, J. Ocumpaugh, and N. Heffernan. Studying affect dynamics and chronometry using sensor-free detectors. In *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[46] C. Cody, M. Maniktala, D. Warren, M. Chi, and T. Barnes. Does autonomy help Help? The impact of unsolicited hints and choice on help avoidance and learning. In *Proceedings of the 13th International Conference on Educational Data Mining*, 2020.

[47] M. Dong, R. Yu, and Z. Pardos. Design and deployment of a better university course search: Inferring latent keywords from enrollments. In *Proceedings of the 12th International Conference on Educational Data Mining*, 2019.

[48] M. Eagle, A. Corbett, J. Stamper, and B. McLaren. Predicting individualized learner models across tutor lessons. In *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[49] E. Farhana, T. Rutherford, and C. Lynch. Investigating relations between self-regulated reading behaviors and science question difficulty. In *Proceedings of the 13th International Conference on Educational Data Mining*, 2020.

[50] S. C. Fonseca, F. D. Pereira, E. Oliveira, D. Fernandes, L. S. D. Carvalho, and A. Cristea. Automatic subject-based contextualisation of programming assignment lists. In *Proceedings of the 13th International Conference on Educational Data Mining*, 2020.

[51] C. Forsyth, J. Andrews-Todd, and J. Steinberg. Are you really a team player? profiling of collaborative problem solvers in an online environment. In *Proceedings of the 13th International Conference on Educational Data Mining*, 2020.

[52] P. S. Inventado, P. Scupelli, E. V. Inwegen, K. S. Ostrow, N. Heffernan, J. Ocumpaugh, R. Baker, S. Slater, and M. Almeda. Hint availability slows completion times in summer work. In *Proceedings of the Ninth International Conference on Educational Data Mining*, 2016.

[53] S. Klingler, R. Wampfler, T. Käser, B. Solenthaler, and M. Gross. Efficient feature embeddings for student classification with variational auto-encoders. In *Proceedings of the 10th International Conference on Educational Data Mining*, 2017.

[54] Z. Liu, R. Brown, C. Lynch, T. Barnes, R. Baker, Y. Bergner, and D. McNamara. MOOC learner behaviors by country and culture; an exploratory analysis. In *Proceedings of the Ninth International Conference on Educational Data Mining*, 2016.

[55] Z. Liu, C. Cody, T. Barnes, C. Lynch, and T. Rutherford. The antecedents of and associations with elective replay in an educational game: Is replay worth it? In *Proceedings of the 10th International Conference on Educational Data Mining*, 2017.

[56] I. Pytlarz, S. Pu, M. Patel, and R. Prabhu. What can we learn from college students' network transactions? Constructing useful features for student success prediction. In *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[57] S. Slater, J. Ocumpaugh, R. Baker, P. Scupelli, P. S. Inventado, and N. Heffernan. Semantic features of math problems: Relationships to student learning and engagement. In *Proceedings of the Ninth International Conference on Educational Data Mining*, 2016.

[58] K. Thaker, Y. Huang, P. Brusilovsky, and H. Da-qing. Dynamic knowledge modeling with heterogeneous activities for adaptive textbooks. In *Proceedings of the*

11th International Conference on Educational Data Mining, 2018.

[59] A. Vail, J. Wiggins, J. F. Grafsgaard, K. Boyer, E. Wiebe, and J. Lester. The affective impact of tutor questions: Predicting frustration and engagement. In *Proceedings of the Ninth International Conference on Educational Data Mining*, 2016.

[60] O. Vainas, Y. B. David, R. Gilad-Bachrach, M. Ronen, O. Bar-Ilan, R. Shillo, G. Lukin, and D. Sitton. Staying in the zone: Sequencing content in classrooms based on the zone of proximal development. In *Proceedings of the 12th International Conference on Educational Data Mining*, 2019.

[61] R. Venant and M. d'Aquin. Towards the prediction of semantic complexity based on concept graphs. In *Proceedings of the 12th International Conference on Educational Data Mining*, 2019.

[62] R. Zhi, S. Marwan, Y. Dong, N. Lytle, T. W. Price, and T. Barnes. Toward data-driven example feedback for novice programming. In *Proceedings of the 12th International Conference on Educational Data Mining*, 2019.

Student Performance Prediction Using Dynamic Neural Models

Marina Delianidi
International Hellenic University, Greece
dmarina@ihu.gr

Konstantinos Diamantaras
International Hellenic University, Greece
kdiamant@ihu.gr

George Chrysogonidis
International Hellenic University, Greece
g.chrysogonidis@ihu.edu.gr

Vasileios Nikiforidis
International Hellenic University, Greece
v.nikiforidis@ihu.edu.gr

ABSTRACT

We address the problem of predicting the correctness of the student's response on the next exam question based on their previous interactions in the course of their learning and evaluation process. We model the student performance as a dynamic problem and compare the two major classes of dynamic neural architectures for its solution, namely the finite-memory Time Delay Neural Networks (TDNN) and the potentially infinite-memory Recurrent Neural Networks (RNN). Since the next response is a function of the knowledge state of the student and this, in turn, is a function of their previous responses and the skills associated with the previous questions, we propose a two-part network architecture. The first part employs a dynamic neural network (either TDNN or RNN) to trace the student knowledge state. The second part applies on top of the dynamic part and it is a multi-layer feed-forward network which completes the classification task of predicting the student response based on our estimate of the student knowledge state. Both input skills and previous responses are encoded using different embeddings. Regarding the skill embeddings we tried two different initialization schemes using (a) random vectors and (b) pretrained vectors matching the textual descriptions of the skills. Our experiments show that the performance of the RNN approach is better compared to the TDNN approach in all datasets that we have used. Also, we show that our RNN architecture outperforms the state-of-the-art models in four out of five datasets. It is worth noting that the TDNN approach also outperforms the state of the art models in four out of five datasets, although it is slightly worse than our proposed RNN approach. Finally, contrary to our expectations, we find that the initialization of skill embeddings using pretrained vectors offers practically no advantage over random initialization.

Keywords

Student performance prediction, Recurrent neural networks, Time-delay neural networks, Dynamic neural models, Knowledge tracing.

Marina Delianidi, Konstantinos Diamantaras, George Chrysogonidis and Vasileios Nikiforidis "Student Performance Prediction Using Dynamic Neural Models". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 46-54. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

1. INTRODUCTION

Knowledge is distinguished by the ability to evolve over time. This progression of knowledge is usually incremental and its formation is related to the cognitive areas being studied. The process of Knowledge Tracing (KT) defined as the task of predicting students' performance has attracted the interest of many researchers in recent decades [4]. The Knowledge State (KS) of a student is the degree of his or her mastering the Knowledge Components (KC) in a certain domain, for example "Algebra" or "Physics". A knowledge component generally refers to a learnable entity, such as a concept or a skill, that can be used alone or in combination with other KCs in order to solve an exercise or a problem [9]. Knowledge Tracing is the process of modeling and assessing a student's KS in order to predict his or her ability to answer the next problem correctly. The estimation of the student's knowledge state is useful for improving the educational process by identifying the level of his/her understanding of the various knowledge components. By exploiting this information it is possible to suggest appropriate educational material to cover the student's weaknesses and thus maximize the learning outcome.

The main problem of Knowledge Tracing is the efficient management of the responses over time. One of the factors which add complexity to the problem of KT is the student-specific learning pace. The knowledge acquisition may differ from person to person and may also be influenced by already existing knowledge. More specifically, KT is predominantly considered as a supervised sequence learning problem where the goal is to predict the probability that a student will answer correctly the future exercises, given his or her history of interactions with previous tests. Thus, the prediction of the correctness of the answer is based on the history of the student's answers in combination with the skill that is currently examined at this time instance.

Mathematically, the KT task is expressed as the probability $P(r_{t+1} = 1|q_{t+1}, X_t)$ that the student will offer the correct response in the next interaction x_{t+1}, where the students learning activities are represented as a sequence of interactions $X_t = \{x_1, x_2, x_3, ..., x_t\}$ over time T. The x_t interaction consists of a tuple (q_t, r_t) which represents the ques-

tion q_t being answered at time t and the student response r_t to the question. Without loss of generality, we shall assume that knowledge components are represented by skills from a set $S = \{s_1, s_2, ..., s_m\}$. One simplifying assumption, used by many authors [24], is that every question in the set $Q = \{q_1, q_2, ..., q_T\}$ is related to a unique skill from S. Then the knowledge levels of the student for each one of the skills in S compose his or her knowledge state.

The dynamic nature of Knowledge Tracing leads to approaches that have the ability to model time-series or sequential data. In this work we propose two dynamic machine learning models that are implemented by time-dependent methods, specifically recurrent and time delay neural networks. Our models outperform the current state-of-the-art approaches in four out of five benchmark datasets that we have studied. The proposed models differ from the existing ones in two main architectural aspects:

- we find that attention does not help improve the performance and therefore we make no use of attention layers
- we experiment with and compare between two different skill embedding types: (a) initialized by pretrained embeddings of the textual descriptions of the skill names using standard methods such as Word2Vec and FastText and (b) randomly initialized embeddings based on skill ids

The rest of the paper is organized as follows. Section 2 reviews the related works on KT and the existing models for student performance prediction. In Section 3 we present our proposed models and describe their architecture and characteristics. The datasets we prepared and used are present in Section 4 while the experiments setup and the results are explained in Section 5. Finally, Section 6 concludes this work and discusses the future works and extensions of the research.

2. RELATED WORKS

The problem of knowledge tracing is dynamic as student knowledge is constantly changing over time. Thus, a variety of methods, highly structured or dynamic, have been proposed to predict students' performance. One of the earlier methods is Bayesian Knowledge Tracing (BKT) [4] which models the problem as a Hidden Markov chain in order to predict the sequence of outcomes for a given learner. The Performance Factors Analysis Model (PFA) [14] proposed to tackle the knowledge tracing task by modifying the Learning Factor Analysis model. It estimates the probability that a student will answer a question correctly by maximizing the likelihood of a logistic regression model. The features used in the PFA model, although interpretable, are relatively simple and designed by hand, and may not adequately represent the students' knowledge state [23].

Deep Knowledge Tracing (DKT) [15] is the first dynamic model proposed in the literature utilizing recurrent neural networks (RNN) and specifically the Long Short-Term Memory (LSTM) model [6] to track student knowledge. It uses one-hot encoded skill tags and associated responses as inputs and it trains the neural network to predict the next student response. The hidden state of the LSTM can be considered as the latent knowledge state of a student and can carry the information of the past interactions to the output layer. The output layer of the model computes the probability of the student answering correctly a question relating to a specific Knowledge Component.

Another approach for predicting student performance is the Dynamic Key-Value Memory Network (DKVMN) [24] which relies on an extension of *memory networks* proposed in [12]. The model tries to capture the relationship between different concepts. The DKVMN model outperforms DKT using memory slots as key and value components to encode the knowledge state of students. Learning or forgetting of a particular skill are stored in those components and controlled by read and write operations through the Least Recently Used Access (LRUA) attention mechanism [16]. The key component is responsible for storing the concepts and is fixed during testing while the value component is updated when a concept state changes. The latter means that when a student acquires a concept in a test the value component is updated based on the correlation between exercises and the corresponding concept.

The Deep-IRT model [23] is the newest approach that extends the DKVMN model. The author combined the capabilities of DKVMN with the Item Response Theory (IRT) [5] in order to measure both student ability and question difficulty. At the same time, another model, named Sequential Key-Value Memory Networks (SKVMN) [1], tried to overcome the problem of DKVMN to capture long term dependencies in the sequences of exercises and generally in sequential data. This model combines the DKVMN mechanism with the Hop-LSTM, a variation of LSTM architecture and has the ability to discover sequential dependencies among exercises, but it skips some LSTM cells to approach previous concepts that are considered relevant. Finally, another newly proposed model is Self Attentive Knowledge Tracing (SAKT) [13]. SAKT utilizes a self-attention mechanism and mainly consists of three layers: an embedding layer for interactions and questions followed by a Multi-Head Attention layer [19] and a feed-forward layer for student response prediction.

The above models either use simple features (e.g. PFA) or they use machine learning approaches such as key-value memory networks or attention mechanisms that may add significant complexity. However we will show that similar and often, in fact, better performance can be achieved by simpler dynamic models combining embeddings and recurrent and/or time-delay feed-forward networks as proposed next.

3. PROPOSED APPROACH

3.1 Dynamic Models

As referenced in the relative literature, knowledge change over time is often modeled by dynamic neural networks. The dynamic models produce output based on a time window, called "context window", that contains the recent history of inputs and/or outputs.

There are two types of dynamic neural networks (Figure 1):

(a) Time-Delay Neural Networks (TDNN), with only feed-forward connections and finite-memory of length L equal to the length of the context window, and (b) Recurrent Neural Networks (RNN) with feed-back connections that can have potentially infinite-memory although, practically, their memory length is dictated by a forgetting factor parameter.

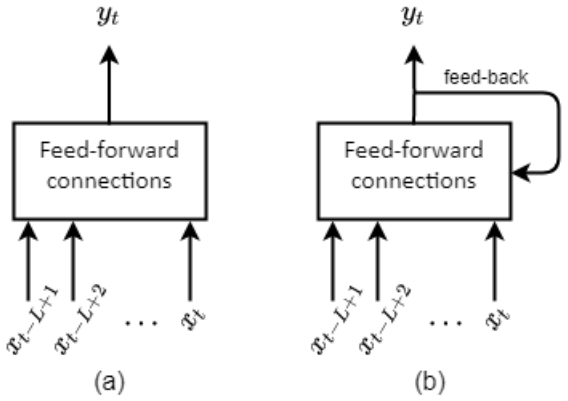


Figure 1: Dynamic model architectures: (a) Time-Delay Neural Network (b) Recurrent Neural Network.

3.2 The Proposed Models

We approach the task of predicting the student response (0=wrong, 1=correct) on a question involving a specific skill as a dynamic binary classification problem. In general, we view the response r_t as a function of the previous student interactions:

$$r_t = h(q_t, q_{t-1}, q_{t-2}, \dots, r_{t-1}, r_{r-2}, \dots) + \epsilon_t \qquad (1)$$

where q_t, is the skill tested on time t and ϵ_t is the prediction error. The response is therefore a function of the current and the previous tested skills $\{q_t, q_{t-1}, q_{t-2}, \dots\}$, as well as the previous responses $\{r_{t-1}, r_{t-2}, \dots\}$ given by the student.

We implement h as a dynamic neural model. Our proposed general architecture is shown in Figure 2. The inputs are the skill and response sequences $\{q\}$, $\{r\}$ collected during a time-window of length L prior to time t. Note that the skill sequence includes the current skill q_t but the response sequence does not contain the current response which is actually what we want to predict. The architecture consists of two main parts:

- The Encoding sub-network. It is used to represent the response and skill input data using different embeddings. Clearly, embeddings are useful for encoding skills since skill ids are categorical variables. We found that using embeddings to encode responses is also very beneficial. The details of the embeddings initialization and usage are described in the next section.
- The Tracing sub-network. This firstly estimates the knowledge state of the student and then uses it to predict his/her response. Our model function consists of two parts: (i) the Knowledge-Tracing part, represented by the dynamic model f, which predicts the student knowledge state $\mathbf{v}_t$ and (ii) the classification part g, which predicts the student response based on the estimated knowledge state:

$$\mathbf{v}_t = f(q_t, q_{t-1}, q_{t-2}, \dots, r_{t-1}, r_{r-2}, \dots) \qquad (2)$$
$$\hat{r}_t = g(\mathbf{v}_t) \qquad (3)$$

Depending on the memory length, we obtain two categories of models:

(a) models based on RNN networks which can potentially have infinite memory. In this case the KT model is recurrent:

$$\mathbf{v}_t = f(\mathbf{v}_{t-1}, q_t, q_{t-1}, \dots, q_{t-L}, r_{t-1}, \dots, r_{r-L})$$

(b) models based on TDNN networks which have finite memory of length L. In this case the KT model has finite impulse response L:

$$\mathbf{v}_t = f(q_t, q_{t-1}, \dots, q_{t-L}, r_{t-1}, \dots, r_{r-L})$$

Although RNNs have been used in the relevant literature, it is noteworthy that TDNN approaches have not been investigated in the context of knowledge tracing. The classification part is modeled by a fully-connected feed-forward network with a single output unit.

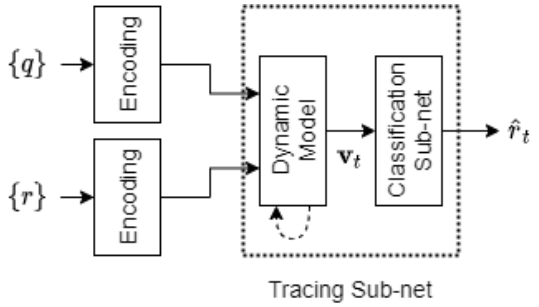


Figure 2: General proposed architecture. The dynamic model can be either a Recurrent Neural Network (with a feedback connection from the output of the dynamic part into the model input) or a Time Delay Neural Network (without feedback connection).

We investigated two different architectures: one based on recurrent neural networks and another based on time delay neural networks. The details of each proposed model architecture are described below.

3.3 Encoding Sub-network

The first part in all our proposed models consists of two parallel embedding layers with dimensions d_q and d_r, respectively, which encode the tested skills and the responses given by the student. During model training the weights of the Embedding layers are updated. The response embedding vectors are initialized randomly. The skill embedding vectors, on the other hand, are initialized either randomly or using pretrained data. In the latter case we use pretrained vectors corresponding to the skill names obtained from Word2Vec [11] or FastText [7] methods.

A 1D spatial dropout layer [18] is added after each Embedding layer. The intuition behind the addition of spatial

dropout was the overfitting phenomenon that was observed in the first epochs of each validation set. We postulated that the correlation among skill name embeddings, that might not actually exist, confused the model.

3.4 Tracing Sub-network

We experimented with two types of main dynamic sub-networks, namely Recurrent Neural Networks and Time Delay Neural Networks. These two approaches are described next.

3.4.1 RNN Approach: Bi-GRU Model

The model architecture based on the RNN method for the knowledge tracing task is shown in Figure 3.

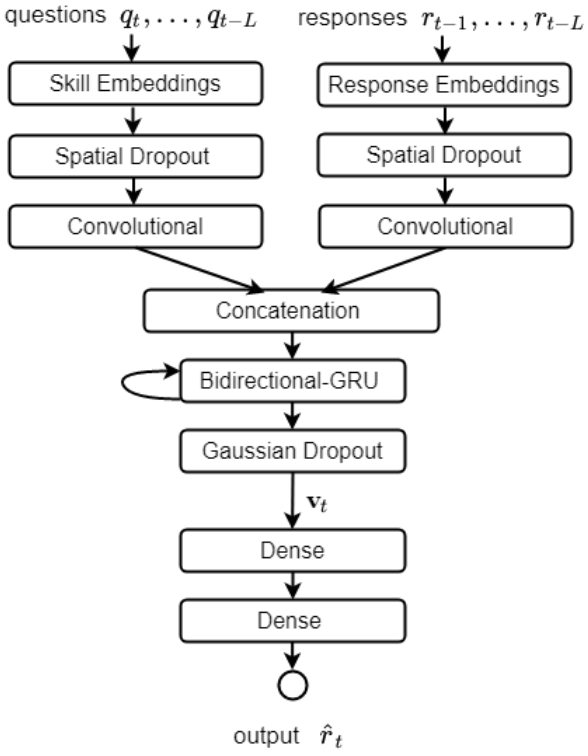


Figure 3: Bi-GRU model

The Spatial Dropout rate following the input embedding layers is 0.2 for most of used datasets. Next, we feed the skills and the responses input branches into a Convolutional layer consisting of 100 filters, with kernel size 3, stride 1, and ReLU activation function. The Convolutional layer acts as a projection mechanism that reduces the input dimensions from the previous Embedding layer. This is found to help alleviate the overfitting problem. To the best of our knowledge, Convolutional layers have not been used in previously proposed neural models for this task. The two input branches are then concatenated to feed a Bidirectional Gated Recurrent Unit (GRU) layer with 64 units [3]. Batch normalization and ReLU activation layers are applied between convolutional and concatenation layers. This structure has resulted after extensive experiments with other popular recurrent models such as LSTM, plain GRU and also bi-directional versions of those models and we found this to be the proposed architecture is the most efficient one.

On top of the RNN layer we append a fully connected sub-network consisting of three dense layers with 50 and 25 units and one output unit respectively. The first two dense layers have a ReLU activation function while the last one has sigmoid activation which is used to make the final prediction ($0 < \hat{r}_t < 1$).

3.4.2 TDNN Approach

In our TDNN model (Figure 4) we add a Convolutional layer after each embedding layer with 50 filters and kernel size equal to 5.

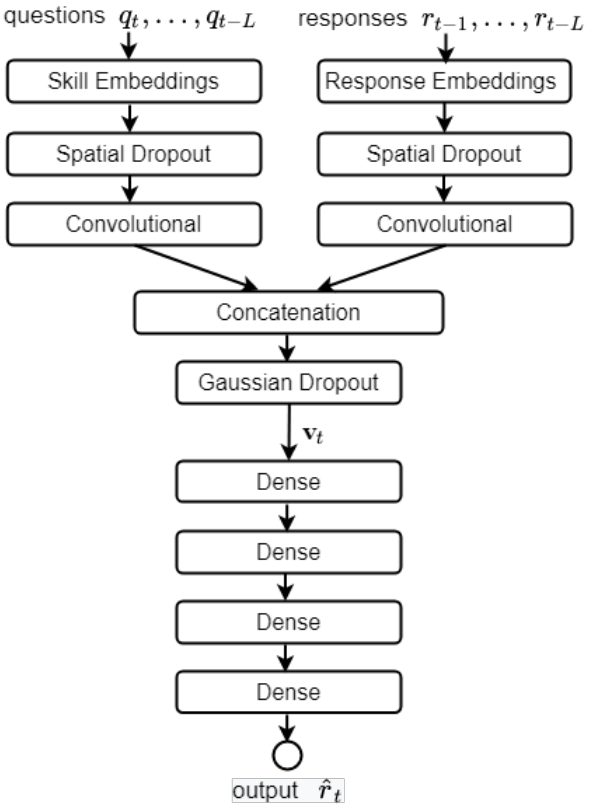


Figure 4: TDNN model

Batch normalization is used before the ReLU activation is applied. As with the RNN model, the two input branches are concatenated to feed the classification sub-network. It consists of four dense layers with 20, 15, 10, and 5 units respectively, using the ReLU activation function. This funnel schema of hidden layers (starting with wider layers and continuing with narrower ones) has helped achieve better results for all datasets we have experimented with. In the beginning of the classification sub-network we insert a Gaussian Dropout layer [17] which multiplies neuron activations with a Gaussian random variable of mean value 1. This has been shown to work as good as the classical Bernoulli noise dropout and in our case even better.

4. DATASETS

We tested our models using four popular datasets from the ASSISTments online tutoring platform. Three of them, "*AS-*

Table 1: Datasets Overview.

Dataset	Skills	Students	Responses	Baseline Accuracy
ASSISTment09	110	4,151	325,637	65.84%
ASSISTment09 corrected	101	4,151	274,590	66.31%
ASSISTment12	196	28,834	2,036,080	69.65%
ASSISTment17	101	1,709	864,713	62.67%
FSAI-F1toF3	99	310	51,283	52.98%

SISTment09", "*ASSISTment09 corrected*"[1], and "*ASSISTment12*"[2] were provided by the above platform. The fourth dataset, named "*ASSISTment17*" was obtained from 2017 Data Mining competition page[3]. Finally a fifth dataset, "*FSAI-F1toF3*" provided by "Find Solution Ai Limited" was also used in our experiments. It is collected using data from the from the 4LittleTrees[4] adaptive learning application.

4.1 Datasets Descriptions

The ASSISTments datasets contain data from student tests on mathematical problems [2] and the content is organized in columns style. The student's interaction is recorded on each line. There are one or more interactions recorded for each student. We take into account the information concerning the responses of students to questions related with a skill. Thus, we use the following columns: "*user_id*", "*skill_id*", "*skill_name*", and "*correct*". The "*skill_name*" contains a verbal description of the skill tested. The "*correct*" column contains the values of the students' responses which are either 1 (for correct) or 0 (for wrong).

The original "*ASSISTment09*" dataset contains 525,534 student responses. It has been used extensively in the KT task from several researchers but according to [2] data quality issues have been detected concerning duplicate rows. In our work we used the "*preprocessed ASSISTment09*" dataset found on DKVMN[5] and Deep-IRT[6] models GitHubs. In this dataset the duplicate rows and the empty field values were cleaned, so that finally 1,451 unique students participate with 325,623 total responses and 110 unique skills.

Even after this cleaning there are still some problems such as duplicate skill ids for the same skill name. These problems have been corrected in the "*Assistment09 corrected*" dataset. This dataset contains 346,860 students interactions and has been recently used in [21].

The "*ASSISTment12*" dataset contains students' data until the school year 2012-2013. The initial dataset contains 6,123,270 responses and 198 skills. Some of the skills have the same skill name but different skill id. The total number of skill ids is 265. The "*Assistment17*" dataset contains 942,816 students responses and 101 skills.

[1]https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010
[2]https://sites.google.com/site/assistmentsdata/home/2012-13-school-data-with-affect
[3]https://sites.google.com/view/assistmentsdatamining/data-mining-competition-2017
[4]https://www.4littletrees.com
[5]https://github.com/jennyzhang0215/DKVMN
[6]https://github.com/ckyeungac/DeepIRT

Finally, the "*FSAI-F1toF3*" dataset is the smallest dataset we used. It involves responses to mathematical problems from 7th grade to 9th grade Hong Kong students and consists of 51,283 students responses from 310 students on 99 skills and 2,266 questions. As it is commonly the case in most studies using this dataset, we have used the question tag as the model input q_t.

4.2 Data Preprocessing

No preprocessing was performed on the "*ASSISTment09*" and "*FSAI-F1toF3*" datasets. For the remaining datasets we followed three preparation steps.

First, the skill ids had been repaired by replacement. In particular, the "*ASSISTments09 corrected*" dataset contained skills of the form of "*skill1_skill2*" and "*skill1_skill2_skill3*" which correspond to the same skill names, so we have merged them into the first skill id, found before the underscore. In other words, the skill "*10_13*" was replaced with skill "*10*" and so on. Moreover, few misspellings were observed that were corrected and the punctuations found in three skill names were converted to the corresponding words. For example, in the skill name "*Parts of a Polnomial Terms Coefficient Monomial Exponent Variable*" we corrected the "*Polnomial*" with "*Polynomial*". Also, in the skill name "*Order of Operations +,-,/,*() positive reals*" we replaced the symbols "*+,-,/,* ()*" with the words that express these symbols, ie. "*addition subtraction division multiplication parentheses*". The latter preprocessing action was preferred over the removal of punctuations since the datasets referred to mathematical methods and operations and without them, we would lose the meaning of each skill. Similar procedure has been followed for the "*ASSISTments12*" dataset. Furthermore, spaces after some skill names were removed i.e. the skill name "*Pattern Finding* " became "*Pattern Finding*". In the "*ASSISTment17*" dataset we came across skill names as "*application: multi-column subtraction*" and corrected them by replacing punctuation marks such as "*application multi column subtraction*". That text preparation operations made to ease the generation of word embeddings of the skill names descriptions. In addition, in the "*ASSISTment17*" dataset, the problem ids are used instead of the skill ids. We had to match and replace the problem ids with the corresponding skill ids with the aim of uniformity of the datasets between them.

Secondly, all rows containing missing values were discarded. Thus, after the preprocessing, the statistics of the data sets were formulated as described in the Table 1.

Finally, we split the datasets so that 70% was used for training and 30% for testing. Then, the training subset was further split into five train-validation subsets using 80% for

training and 20% for validation.

5. EXPERIMENTS

In this section we experimentally validate the effectiveness of the proposed methods by comparing them with each other and also with other state-of-the-art performance prediction models. The Area Under the ROC Curve (AUC) [10] metric is used for comparing the predicting probability correctness of student's response.

The state-of-the-art knowledge tracing models we are compared with the DKT, DKVMN and Deep-IRT. We performed the experiments for our proposed models Bi-GRU, TDNN as well as for each of the previous model for all datasets, using the code provided by the authors on their GitHubs. It is worth noting that the python GitHub code[7] used for the DKT model experiments requires the entire dataset file and the train/test splitting is performed during the code execution.

All the experiments were performed on a workstation with Ubuntu operating system, Intel i5 CPU and 16GB Titan Xp GPU card.

5.1 Skill embeddings initialization

As mentioned earlier, skill embeddings are initialized either randomly or using pretrained vectors. Regarding the initialization of the skill embeddings with pretrained vectors we used two methods described next. In first method we used the text files from Wikipedia2Vec[8] [22] that is based on Word2Vec method and contains pretrainable embeddings for the word representation vectors in English language in 100 and 300 dimensions. In second method we used the "*SISTER*" (SImple SenTence EmbeddeR)[9] library to prepare the skill name embeddings based on FastText in 300 dimensions pretrained word embeddings. Each skill name consists of one or more words. Thus, for the Word2Vec method, the skill name embeddings vector is created by adding the word embeddings vectors, while in case of FastText, the skill name embeddings are created by taking the average of the word embeddings.

Especially for the FsaiF1toF3 dataset, the question embeddings are initialized either randomly or using the pretrained word representations of the corresponding skill descriptions by employing the Wikipedia2Vec and SISTER methods as described above. Since many questions belong to the same skill, in this case the corresponding rows in the embedding matrix are initialized by the same vector.

5.2 Experimental Settings

We performed the cross-validation method for the 5 training and validation set pairs. This was to choose the best architecture and parameter settings for each of the proposed models. Using the train and test sets we evaluated the chosen architectures for all the datasets.

[7]https://github.com/lccasagrande/Deep-Knowledge-Tracing

[8]https://wikipedia2vec.github.io/wikipedia2vec/

[9]https://pypi.org/project/sister/

One of the basic hyperparameters of our models that affect to the inputs is the L. It represents the student's interaction history window length. The inputs with L sequence of questions and $L-1$ sequence of responses. The best results we succeeded are when using $L = 50$ for the both Bi-GRU and TDNN models. The batch sizes used in the models during the training are: 32 in Bi-GRU and 50 in TDNN.

Since specific dimensions of the pretrained word embeddings are provided, we used the same dimensions in case of random embedding in order to take the comparable results. Skill embeddings and responses embeddings set in the same dimensions.

The scheduler learning rate is implementing in Bi-GRU starting from 0.001 and reducing over the training operation of the models that performs for 30 epochs. During training we applied the following learning rate schedule depending on the epoch number n:

$$lr = \begin{cases} r_{init} & \text{if } n < 10 \\ r_{init} \times e^{(0.1 \cdot (10-n))} & \text{otherwise} \end{cases}$$

In case of the TDNN-based model, the learning rate equals 0.001 and is the same during the whole training process for 30 epochs. We used cross-entropy optimization criterion and the Adam or AdaMax [8] learning algorithms.

Dropout with rate = 0.2 or 0.9 is also applied to the Bi-GRU model while the dropout rate of the TDNN equals to one of the $(0.2, 0.4, 0.6, 0.9)$ values through to the Gaussian dropout layer. We observed a reduction of overfitting during model training by changing the Gaussian dropout rate relative to the dataset's size. Thus, the smaller dataset size is, the bigger dropout rate has been used.

The various combinations of parameters settings were applied during the experimental process for all proposed models presented in Table 2.

5.3 Experimental Results

The experiments results of our models are shown in Table 3. Comparing our models with each other we can see that the RNN-based Bi-GRU model outperforms the TDNN-based model in all datasets. It achieved best results when 100d embeddings were used either in pretrained or the random initialization type.

We observed that in both Bi-GRU or TDNN, the embedding type is not the significant parameter that affects the models performance. The differences between the results of the experiments showed that the size of embeddings dimensions not particularly contributed to the final result and the difference in performance of the models was small.

Except for our models, we performed experiments for all datasets on the previous models we compared. For three of the datasets, specifically for "*ASSISTment09 corrected*", "*ASSISTment12*" and "*ASSISTment17*" there were not available results in the corresponding papers. In this paper, we present the results of the experiments we run using that models codes.

Table 2: Models experiments settings

Parameters	Bi-GRU	TDNN
Learning rate	0.001	0.001
Learning rate schedule	yes	no
Training epochs	30	30
Batch size	32	50
Optimizer	Adam	AdaMax
History window length	50	50
Skill embeddings dim.	100 & 300	100 & 300
Skill embeddings type	Random, W2V, FastText	Random, W2V, FastText
Responses embeddings dim.	Same to skill dim.	Same to skill dim.
Responses embeddings type	Random	Random

Table 3: Comparison between our proposed models - AUC (%). (R) = random skill embedding initialization, (W) = skill embedding initialization using W2V, (F) = skill embedding initialization using FastText. Datasets: (a) *ASSISTment09*, (b) *ASSISTment09 corrected*, (c) *ASSISTment12*, (d) *ASSISTment17*, (e) *FSAI-F1toF3*

	$d_q = 100$(R)	$d_q = 300$(R)	$d_q = 100$(W)	$d_q = 300$(W)	$d_q = 300$(F)
Bi-GRU	82.55	82.45	82.52	82.55	82.39
TDNN	81.54	81.67	81.59	81.50	81.53

(a)

	$d_q = 100$(R)	$d_q = 300$(R)	$d_q = 100$(W)	$d_q = 300$(W)	$d_q = 300$(F)
Bi-GRU	75.27	75.13	75.14	75.09	75.12
TDNN	74.38	74.39	74.40	74.33	74.37

(b)

	$d_q = 100$(R)	$d_q = 300$(R)	$d_q = 100$(W)	$d_q = 300$(W)	$d_q = 300$(F)
Bi-GRU	68.37	68.37	68.40	68.23	68.27
TDNN	67.95	67.97	67.99	67.95	67.91

(c)

	$d_q = 100$(R)	$d_q = 300$(R)	$d_q = 100$(W)	$d_q = 300$(W)	$d_q = 300$(F)
Bi-GRU	73.62	73.58	73.76	73.54	73.58
TDNN	71.68	71.75	71.52	71.81	71.83

(d)

	$d_q = 100$(R)	$d_q = 300$(R)	$d_q = 100$(W)	$d_q = 300$(W)	$d_q = 300$(F)
Bi-GRU	70.47	69.34	70.24	69.80	69.51
TDNN	70.03	69.80	69.80	70.11	70.06

(e)

The best experimental results of the ours models in comparison with the previous models for each dataset are presented in Table 4. The model that has the best performance for the four of datasets is the Bi-GRU. Except for that, the TDNN-based model has better performance in comparison to the previous models for four datasets. The only dataset, for which the previous models overcomed our models is the "*ASSISTment12*".

5.4 Discussion

Our model architecture is loosely based on the DKT model and offers improvements in the aspects discussed below. First, we employ embeddings for representing both skills and responses. It is known that embeddings offer more useful representations compared to one-hot encoding because they can capture the similarity between the items they represent [20]. Second, we thoroughly examined dynamical neural models for estimating the student knowledge state by trying both infinite-memory RNNs and finite-memory TDNNs. To our knowledge, TDNNs have not been well studied in the literature with respect to this problem. Third, we used convolutional layers in the inputs encoding sub-net. We found that this layer functioned as a reducing mechanism of the embedding dimensions and in conjunction with the dropout layer mitigated the overfitting problem. The use of Convolutional layers is a novelty in models tackling the knowledge tracing problem. Fourth, unlike DKT, we used more hidden layers in the classification sub-net. Our experiments demonstrate that this gives more discriminating capability to the classifier and improves the results. Finally, our experiments with key-value modules and attention mechanism did not help further improve our results and so these experiments are not reported here. In the majority of the datasets we examined our model outperforms the state-off the models employing key-value mechanisms such as DKVMN and Deep-IRT.

In addition to the AUC metric which is typically used for

Table 4: Comparison test results of evaluation measures - the AUC metric (%)

Dataset	DKT	DKVMN	Deep-IRT	Bi-GRU	TDNN
ASSISTment09	81.56%	81.61%	81.65%	**82.55**%$^{(1,2)}$	81.67%$^{(3)}$
ASSISTment09 corrected	74.27%	74.06%	73.41%	**75.27**%$^{(1)}$	74.40%$^{(2)}$
ASSISTment12	69.40%	69.26%	**69.73**%	68.40%$^{(4)}$	67.99%$^{(4)}$
ASSISTment17	66.85%	70.25%	70.54%	**73.76**%$^{(4)}$	71.83%$^{(5)}$
FSAI-F1toF3	69.42%	68.40%	68.69%	**70.47**%$^{(1)}$	70.11%$^{(2)}$

$^{(1)}$ $d_q = d_r = 100$, Random, $^{(2)}$ $d_q = d_r = 300$, W2V, $^{(3)}$ $d_q = d_r = 300$, Random, $^{(4)}$ $d_q = d_r = 100$, W2V, $^{(5)}$ $d_q = d_r = 300$, FastText

Table 5: Statistical significance testing results of Bi-GRU and TDNN

Dataset	P-value
ASSISTment09	7.34 e-59
ASSISTment09 corrected	2.31 e-52
ASSISTment12	1.45 e-203
ASSISTment17	7.96 e-44
FSAI-F1toF3	1.38 e-84

evaluating the performance of our machine learning models, we applied statistical significance testing to check the similarity between out Bi-GRU and TDNN models. Specifically, we performed a T-Test between the outcomes of the two models in all training data using the best configuration settings as shown in Table 4. The results reported in Table 5 show that the P-value calculated in all cases is practically zero which proves the hypothesis that the two models are significantly different.

6. CONCLUSION AND FUTURE WORK

In this paper we propose a novel two-part neural network architecture for predicting student performance in the next exam or exercise based on their performance in previous exercises. The first part of the model is a dynamic network which tracks the student knowledge state and the second part is a multi-layer neural network classifier. For the dynamic part we tested two different models: a potentially infinite memory recurrent Bidirectional GRU model and a finite memory Time-Delay neural network (TDNN). The experimental process showed that the Bi-GRU model achieves better performance compared to the TDNN model. Despite the fact that TDNN models have not been used for this problem in the past, our results have shown that they can be just as efficient or even better compared to previous state-of-art RNN models and only slightly worse than our proposed RNN model. The model inputs are the student's skills and responses history which are encoded using embedding vectors. Skill embeddings are initialized either randomly or by pretrained vectors representing the textual descriptions of the skills. A novel feature of our architecture is the addition of spatial dropout and convolutional layers immediately after the embeddings layers. These additions have been shown to reduce the overfitting problem. We found that the choice of initialization of the skill embeddings has little effect on the outcome of our experiments. Moreover, noting that there is a different use of the same datasets in different studies, we described in detail the process of the datasets pre-processing, and we provide the train, validation and test splits of the data that were used in our experiments on our GitHub repository[10]. The extensive experimentation with more benchmark datasets as well as the study of variants of the proposed models will be the subject of our future work with the aim of even further improving the prediction performance of the models.

7. ACKNOWLEDGMENTS

We would like to thank NVIDIA Corporation for the kind donation of an Titan Xp GPU card that was used to run our experiments.

8. REFERENCES

[1] G. Abdelrahman and Q. Wang. Knowledge tracing with sequential key-value memory networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 175–184, 2019. https://doi.org/10.1145/3331184.3331195.

[2] AssistmentsData. Assistments data, 2015. https://sites.google.com/site/assistmentsdata/.

[3] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[4] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[5] R. K. Hambleton, H. Swaminathan, and J. H. Rogers. *Fundamentals of item response theory*. Sage Publications, Newbury Park, CA, USA, 1991.

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov. Fasttext.zip: Compressing

[10] https://github.com/delmarin35/Dynamic-Neural-Models-for-Knowledge-Tracing

text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] K. R. Koedinger, A. T. Corbett, and C. Perfetti. The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science*, 36(5):757–798, 2012.

[10] C. X. Ling, J. Huang, H. Zhang, et al. Auc: a statistically consistent and more discriminating measure than accuracy. In *Ijcai*, volume 3, pages 519–524, 2003.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[12] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

[13] S. Pandey and G. Karypis. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*, 2019.

[14] P. I. J. Pavlik, C. Hao, and K. R. Kenneth. Performance factors analysis–a new alternative to knowledge tracing. In *Proceedings 14th Int. Conf. Artificial Intelligence in Education*, Brighton, England, 2009. ERIC.

[15] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. *Advances in neural information processing systems*, 28:505–513, 2015.

[16] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[18] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 648–656, 2015.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[20] S. Wang, W. Zhou, and C. Jiang. A survey of word embeddings based on deep learning. *Computing*, 102(3):717–740, 2020.

[21] L. Xu and M. A. Davenport. Dynamic knowledge embedding and tracing. *arXiv preprint arXiv:2005.09109*, 2020.

[22] I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, and Y. Matsumoto. Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 23–30. Association for Computational Linguistics, 2020. https://wikipedia2vec.github.io/wikipedia2vec/.

[23] C.-K. Yeung. Deep-irt: Make deep learning based knowledge tracing explainable using item response theory. *arXiv preprint arXiv:1904.11738*, 2019.

[24] J. Zhang, X. Shi, I. King, and D.-Y. Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*, pages 765–774, 2017.

Say What? Automatic Modeling of Collaborative Problem Solving Skills from Student Speech in the Wild

Samuel L. Pugh[1], Shree Krishna Subburaj[1], Arjun Ramesh Rao[1], Angela E.B. Stewart[2], Jessica Andrews-Todd[3], Sidney K. D'Mello[1]
[1]University of Colorado Boulder; [2]Carnegie Mellon University; [3]Educational Testing Service
samuel.pugh@colorado.edu; jandrewstodd@ets.org; sidney.dmello@colorado.edu

ABSTRACT
We investigated the feasibility of using automatic speech recognition (ASR) and natural language processing (NLP) to classify collaborative problem solving (CPS) skills from recorded speech in noisy environments. We analyzed data from 44 dyads of middle and high school students who used videoconferencing to collaboratively solve physics and math problems (35 and 9 dyads in school and lab environments, respectively). Trained coders identified seven cognitive and social CPS skills (e.g., sharing information) in 8,660 utterances. We used a state-of-the-art deep transfer learning approach for NLP, Bidirectional Encoder Representations from Transformers (BERT), with a special input representation enabling the model to analyze adjacent utterances for contextual cues. We achieved a micro-average AUROC score (across seven CPS skills) of .80 using ASR transcripts, compared to .91 for human transcripts, indicating a decrease in performance attributable to ASR error. We found that the noisy school setting introduced additional ASR error, which reduced model performance (micro-average AUROC of .78) compared to the lab (AUROC = .83). We discuss implications for real-time CPS assessment and support in schools.

Keywords
Collaborative problem solving; natural language processing; collaborative interfaces

1. INTRODUCTION
The modern world will increasingly require teams of heterogeneous individuals to coordinate their efforts, share skills and knowledge, and communicate effectively in order to solve complex and pressing problems like the global pandemic and climate change. Accordingly, collaborative problem solving (CPS) – defined as two or more people engaging in a coordinated attempt to construct and maintain a joint solution to a problem [57] – has been identified as a critical skill for the 21st century workforce [23, 27]. Despite its increasing importance, the most recent 2015 Programme for International Student Assessment (PISA) assessment revealed troubling deficiencies in CPS competency worldwide [49]. As a result, improving CPS proficiency has become a priority in educational research and policy [7, 8, 16, 37, 49].

Samuel Pugh, Shree Krishna Subburaj, Arjun Ramesh Rao, Angela Stewart, Jessica Andrews-Todd and Sidney D'Mello "Say What? Automatic Modeling of Collaborative Problem Solving Skills from Student Speech in the Wild". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 55-67. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Technology has fundamentally transformed both the modern workplace and classroom. Co-located teams in shared spaces are becoming less common, while distributed teams that work and collaborate remotely through virtual interfaces are on the rise [22, 36]. In 2020, the COVID-19 pandemic thrust this issue to the forefront of our attention, as workers and students across the globe were forced to adapt to a remote environment for extended periods of time. Accordingly, educational practitioners have emphasized the importance of providing students with the skills necessary to effectively collaborate in virtual settings [60].

The rise of videoconferencing in both workplace and learning environments brings with it the exciting opportunity to develop next-generation collaborative interfaces that can aid in teaching, assessing, and supporting CPS. Here we focus on the task of assessing CPS skills from spoken language with an eye for downstream applications including reflective feedback and dynamic interventions to improve CPS skills.

Like any latent construct (e.g., intelligence, knowledge), assessment of CPS skills entails identifying objective evidence for those constructs. Because collaboration inherently involves communication, one promising approach is to analyze communication between team members [58]. Indeed, the content of communication during CPS provides information about a team's cognitive and affective states, knowledge, information sharing, and coordination [27], and can serve as evidence of relevant CPS skills [3, 4].

However, analyzing the large amounts of data generated during open-ended collaboration is time consuming and costly, requiring trained human coders to review large corpus and hand code individual items for indicators of CPS. Previous work [24, 29, 58, 65] has attempted to automate this coding process using natural language processing (NLP) techniques. However, with the exception of [65], this has been limited to restricted forms of communication such as text chat, rather than open-ended verbal communication, which is characteristic of most real world CPS. As we elaborate below, the one study [65] that successfully analyzed spoken communications for evidence of CPS skills used data collected in a highly controlled lab environment, leaving open the question as to whether this approach will succeed in the wild, such as in noisy classroom environments.

In this work, we address the challenge of using speech recognition and NLP to automatically analyze open-ended student speech during videoconferencing-enabled collaborative problem solving in both real-world schools and in lab environments. Pursuing technologies capable of automatically capturing and analyzing spoken language during open-ended verbal CPS in authentic environments, whether face-to-face or via videoconferencing, is an important avenue of research. These technologies hold the potential for significantly improving real-

time assessment and support of CPS [58], whether by providing teachers with feedback on CPS in student groups or enabling just-in-time interventions to steer groups of problem solvers in the right direction.

1.1 Background and Related Work

We first present a brief discussion on theoretical frameworks of CPS to situate the CPS skills modeled in this study within the CPS literature. Then, we discuss prior work on computational models of CPS, specifically focusing on language-based models.

1.1.1 Frameworks of CPS

CPS has been defined as problem solving activities that involve interactions among a group of individuals [47]. One early attempt to conceptualize CPS was by Roschelle and Teasley [57] who proposed a joint problem space model that emphasized shared understanding of the task as a central aspect of CPS. More recently, the Assessment and Teaching of Twenty-First Century Skills (ATC21S) framework [28, 30] described CPS through a measurable and teachable set of social and cognitive skills based on interaction, self-evaluation and goal setting. Relatedly, the PISA 2015 [49] framework conceptualized CPS as a complex process involving three collaborative dimensions that overlap with four problem-solving processes resulting in 12 CPS skills. Building on these frameworks, Sun et al. [68] proposed a generalized competency framework for CPS skills based on interactions among triads, which defines a hierarchical CPS model involving three high-level facets of CPS, each composed of sub-facets and associated behavioral indicators. Another approach, and the framework adopted in this work, is the in-task assessment framework [34]. Informed by principles of evidence-centered design [41], this framework characterizes CPS through a hierarchical ontology [3], which lays out theoretically-grounded, generalizable CPS skills along with behavioral indicators of these skills.

1.1.2 Computational Models of CPS

The stream of interactions generated during problem solving is considered the richest source of information about a team's knowledge, skills, and abilities [27, 38]. Accordingly, prior research has used non-verbal behavioral signals like facial expressions to detect rapport loss in small groups during open-ended discussions [43]. Multimodal combinations of facial expressions, acoustics and prosody, eye gaze, and task context have been explored to predict CPS outcomes like task performance [42, 67]. Additionally, learning gains [32, 50], subjective performance [72] and CPS competence [13, 14] have been modelled using multimodal signals.

Focusing our review on studies that explored the use of language and speech based data, researchers have successfully used language to model CPS processes like idea sharing [24, 29], negotiation [65], and argumentation [58], as well as CPS outcomes such as task performance [10, 44, 51] and learning gains [55]. A common NLP approach involves quantifying the frequency of words and word phrases (n-grams) [24, 29, 44, 54, 58]. Further, some research has experimented with the use of additional lexical features like punctuation [24, 29, 58], part-of-speech tags [21, 44, 58], or emoticons [29]. In addition to using lexical features from language itself, researchers have derived features from conversational data which index team and conversational dynamics (e.g., turn taking). This approach has been used to provide feedback on collaboration [59], identify sociocognitive roles [20], and model intra- and interpersonal dynamics [19] during CPS.

Closely related to our work, Hao et al. [29] used pre-selected n-grams and emoticons to model four CPS facets of sharing ideas, negotiating, regulating problem-solving activities, and maintaining communication. Their study involved data collected from 1000 participants with at least one year of college experience randomly grouped into dyads. They used a linear chain conditional random field and extracted lexical features from sequential text chats between dyads. They found that sequential modeling achieved an average accuracy of 73.2%, which outperformed a majority-class baseline accuracy of 29%, and slightly outperformed standard classifiers (accuracies of 66.9% to 71.9%).

Whereas the Hao study analyzed text-chats among dyads, Stewart et al. [65] modeled the three CPS facets of construction of shared knowledge, negotiation and coordination, and maintaining team function from spoken trialogues (conversations among triads). The study involved 32 triads of undergraduate students from a medium-sized private university, engaged in a 20-minute computer programming task using video conferencing software in a lab setting. They used ASR to generate transcripts of the team's speech during problem solving, from which they derived n-gram features for modeling. They obtained area under the receiver operating characteristic curve (AUROC) scores of .85, .77 and .77 for the three CPS facets using random forest classifiers, exceeding chance baselines of 0.5. In a follow-up study [66], they investigated whether including additional modalities (facial expression, acoustic-prosodic features, task context) in addition to language improved classification accuracy. They found that a combination of language and task context yielded slight improvement over unimodal language models.

1.2 Current Study and Novelty

There are several novel aspects of this work. First, although recent work [65, 66] has successfully used ASR and NLP to automatically analyze speech during CPS in the lab, it is currently unknown whether this approach can be effective in the wild, for example in noisy real-world classrooms where CPS interactions would occur. Lab environments have the advantage of being free from ambient noises, distractions from other students, and various other complicating factors present in school environments.

Further, previous work has been limited to adults, namely undergraduate students. However, given the importance of CPS, it is imperative that technologies be developed that can help instruct and support CPS in middle and high school-aged students. Therefore, a second important question is whether this approach can be applied to children, who may have differing CPS abilities and communication styles. An accompanying question is whether ASR can provide sufficiently accurate transcripts of children's speech, as research has documented the degradation of ASR performance on children's speech due to ASR systems primarily trained on adult speech, and age-dependent spectral and temporal variability in speech signals [26, 45, 53].

We address these questions by recording audio of remote CPS among middle and high school students in both the lab and

computer-enabled classrooms with multiple teams interacting. We show for the first time that in noisy school environments, ASR can provide transcripts of sufficient accuracy to model CPS skills. Additionally, we quantify the decrease in predictive accuracy that can be attributed to ASR error (vs. NLP error) by comparing with models trained on human transcripts, and comparing lab- vs. classroom- environments.

Finally, an open question in this domain is which NLP algorithms should be used to automatically analyze CPS language. We explore the use of deep transfer learning for this NLP problem. Recent advances in state-of-the-art NLP have been attained by adapting attention-based language models [71], pre-trained on large amounts of unlabeled data, to specific NLP tasks (e.g., text classification) [31]. We demonstrate the efficacy of this approach, using the popular Bidirectional Encoder Representations from Transformers (BERT) model [18] for our NLP task, and compare results with a more traditional n-gram approach using random forest classifiers. We also investigate whether a sequential classifier, which considers adjacent (i.e. previous, subsequent) utterances for contextual cues, yields improved performance over single utterance classifiers. We present a method, similar to the approaches used in [12, 69], to capture adjacent utterances for context by constructing a special input representation for the BERT model, which improves classification accuracy.

2. METHOD

2.1 Data Collection

2.1.1 Contexts

Our primary data collection occurred in one United States east coast public middle school and one public high school from the same district. The study was run over two data collection periods. The first period included 61 students in the high school and 44 students in the middle school. Here, students participated in two 43 minute class periods. The second collection included 18 students from the same middle school. Because we did not have control over the acoustic environment in the school context, we also collected supplementary data from 18 students in the lab. In the second collection, students completed one 90 minute session. In both collections, students in the school environment completed the study from a computer lab in the school in which other students were also participating in the study. Data collection occurred prior to the COVID-19 pandemic, and as such classrooms were at normal capacity. Students in both environments were equipped with a personal headset and microphone (MPOW 071 USB Headset).

2.1.2 Participants

In all, 141 middle and high school students (age range: 12-15) completed some or all of the study. However, only a subset of 74 sessions (a session entails one dyad completing one of the tasks) were included in this analysis. Participants were excluded for the following reasons: we experienced technical challenges on the first day of data collection, either team member did not complete a consent form, one team member did not show up, or there were quality issues with the recorded audio stream. Our analyzed dataset consisted of 88 students (65% female; mean age = 13.6, SD = 0.90). The lab subset contained 18 students (50% female; mean age = 13.6, SD = 1.01) and the school subset contained 70 students (69% female; mean age = 13.6, SD = 0.87). The sample of 88 students was quite diverse with 26.1% self-reporting as Black/African American, 19.3% Hispanic/Latino, 15.9% Multiracial, 13.6% Asian/Asian American, 12.5% White, 2.3% American Indian/Alaska Native, 6.8% reported "Other", and 3.4% did not report ethnicity.

2.1.3 CPS Tasks

The study involved two separate CPS tasks. In one task on linear functions and argumentation (T-Shirt Math Task [1]), students worked together through a series of task items in which they sought to determine which of three t-shirt companies was the best choice for a student council to purchase t-shirts for classmates. They compared three companies with differing variable costs (price per shirt) and fixed costs (upfront fee) to determine which company should be chosen given the number of t-shirts to be purchased. Individual questions included populating the cost equation y = mx + b according to the costs of each company (see Figure 1B), identifying the correct graph for a given company's cost equation, and providing a recommendation as to which company was the best deal. During this task, only one student controlled the screen at a time (i.e. to enter responses to the questions), and the two students could alternate control as they chose.

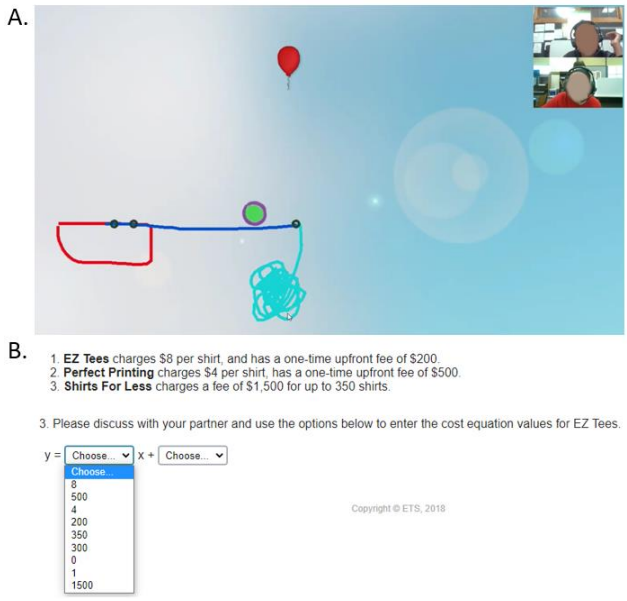


Figure 1. Screenshot examples of the videoconferencing setup and two CPS tasks. (A) Shows a level in Physics Playground, (B) shows a question from the T-Shirt Math Task (reproduced with permission from ETS).

The second task (Physics Playground [62]) was an educational physics game designed to help students learn concepts in Newtonian physics. In this task, students completed a series of six game levels in which they were tasked with drawing objects (e.g., lever, ramp, springboard) to guide a ball to hit a balloon target (see Figure 1A in which students are drawing a weight attached to the springboard to launch the ball towards the balloon). During this task, only one student controlled the game at a time. One student was selected to control first, and after

three levels had been completed (or half of the allotted time had elapsed), control was switched to the other student for the following three levels. Whereas the math task resembles more traditional school work and is more constrained by prior knowledge, the physics game provides more opportunities for creative exploration [35].

2.1.4 Procedure

Students were randomly assigned to pairs (27 mixed-gender, 17 same-gender pairs) and each student first individually completed a series of pre-surveys; details are not relevant here. Once both students in the pair completed the pre-surveys, a researcher enabled audio and video recording on each student's computer using Zoom video conferencing software (https://zoom.us) to record students' computer screens, faces, and voices. The student teams then worked together to complete the two CPS tasks, either on a different day or the same day (see above). The order of the tasks was counterbalanced so that half of the teams completed Physics Playground first and the other half completed the T-Shirt Math Task first. After completing each task students individually completed additional questionnaires not analyzed here.

2.2 CPS Ontology and CPS Skills

2.2.1 CPS Ontology (Framework)

We used a competency model represented as an ontology [3, 4] (similar to a concept map), which lays out the components of CPS and their relationships, along with indicators of CPS skills. The development of the ontology was based on discussions with subject matter experts as well as a literature review in relevant areas such as computer-supported collaborative learning, individual problem solving, communication, and linguistics [30, 39, 46, 48, 49, 64].

Our CPS ontology [3] includes nine high-level CPS skills across social and cognitive dimensions and sub-skills that correspond to each high-level skill. The social dimension includes four CPS skills: (1) *Maintaining communication* corresponds to content irrelevant social communications among teammates (e.g., greeting teammates or engaging in off-topic conversations); (2) *Sharing information* corresponds to task-relevant communication that is useful for solving the problem (e.g., sharing one's own knowledge, sharing the state of one's understanding); (3) *Establishing shared understanding* includes communication used to learn the perspectives of others and ensure that what has been said is understood by teammates (e.g., requesting information from teammates, providing responses that indicate comprehension); and (4) *Negotiating* corresponds to communication used to express agreement, express disagreement, or resolve conflicts that arise.

The cognitive dimension includes five CPS skills: (1) *Exploring and understanding* corresponds to communication and actions used to explore the environments in which teammates are working or understand the problem at hand (e.g., rereading problem prompts); (2) *Representing and formulating* includes communication used to build a mental representation of the problem and formulate hypotheses; (3) *Planning* corresponds to communication used to develop a plan for solving the problem (e.g., determining goals or establishing steps for carrying out a plan); (4) *Executing* corresponds to actions and communication used to carry out a plan (e.g., taking steps to carry out a plan, reporting to teammates what steps you are taking, or making suggestions to teammates about what steps they should take to carry out the plan); and (5) *Monitoring* includes communication used to monitor progress towards the goal or monitor teammates (e.g., checking the progress or status of teammates).

Table 1. The 7 CPS skills modeled, ordered from highest to lowest prevalence

CPS Skill	Base Rate	Dimension	Example Human Transcript	Corresponding ASR Transcript
Sharing Information	.26	Social	(Math) "Okay so first I think we should create like three equations to for each company"	"Okay Sir thank first we should create like three D creations for each arm company"
Establishing Shared Understanding	.25	Social	(Math) "Which one do you think is the best one"	"Twenty it's the best"
Negotiating	.16	Social	(Physics) "Umm no let's just do another idea I don't think it's gonna work anymore"	"Let's just do it another day I don't think it's going to work anymore"
Executing	.14	Cognitive	(Physics) "Okay and now put a weight down on that"	"Okay and now put a weight down on the"
Maintaining Communication	.07	Social	(Physics) "(laughs) Oh no this game is funny bro yeah I don't know what to do"	"This came funny I would like to do"
Monitoring	.06	Cognitive	(Physics) "That didn't work oh no"	"That didn't recall about"
Planning	.05	Cognitive	(Math) "Alright now we have to find a graph for this one now"	"Now we have to find a crusher this one now"

2.2.2 CPS Coding

Video recordings of student task sessions were segmented at the turn (or utterance) level and then coded by three trained raters using Dedoose qualitative analysis software [17]. For the coding, raters viewed each turn for each individual in a team and then labeled the turn as one of the CPS skills from the CPS ontology. To establish reliability, the three trained raters triple coded 20% of the videos. Intraclass correlations (ICCs) were used to estimate interrater reliability across rater judgments, as it can provide information about the consistency of the judgments among raters. The median ICC across the CPS skill ratings was .93, corresponding to excellent agreement [11].

Once reliability was established, the remaining videos were split among the three raters and coded independently. A total of 10,239 turns were coded across 80 CPS sessions with an average of 128 turns per session (SD = 70.5). Two CPS skills (exploring and understanding, and representing and formulating) occurred very infrequently (base rate < 1%) and were excluded from our analysis. The remaining seven CPS skills, with their base rate, cognitive/social dimension, and a sample utterance from the dataset, are shown in Table 1.

2.3 ASR and Human Transcript Generation

After segmenting and coding each utterance, we used the IBM Watson speech-to-text service [33] to generate ASR transcripts for each video. The service outputs transcripts with word-level start and stop times, as well as word-level confidence (between 0 and 1) for each word recognized. We constructed the transcript for each coded utterance by concatenating transcribed words within the utterance's human segmented time window. The confidence for each utterance was computed by taking the mean word confidence over all words in the utterance transcript. Utterances in which no words were recognized were assigned a confidence of 0. Because a single audio stream of each session was recorded (rather than individual audio streams from each student), the ASR transcripts can contain words from both speakers if there was overlap (elaborated below).

We also manually transcribed each utterance from the CPS videos. Human transcribers viewed the video segment (with audio) of each coded utterance and transcribed the words spoken by the indicated speaker (each utterance was coded for an individual student). Speech from the other student, if present in the segment, was not transcribed. Prior to transcription, guidelines were established among the human transcribers to ensure consistency in transcribing informal words or phrases (e.g., gonna, c'mon).

Because the segmented utterances sometimes contained speech from both speakers, we had alignment inconsistencies, as the ASR transcribed all words in a segment while the human transcripts only contained words spoken by the indicated student. To better assess ASR accuracy, we randomly sampled 10 utterances from each CPS session (8.5% of the data) and re-transcribed the utterances to include all words spoken in the segment, regardless of speaker. We refer to this as the Human Transcript Subset. We then computed a word error rate (WER) [9] for each utterance in this subset defined as (substitutions + insertions + deletions) / (words in human transcript), using the python package Jiwer [70].

2.4 Analyzed Dataset

Our dataset contains 74 CPS task sessions from 44 teams. This includes 30 teams with both the math and physics tasks in the dataset, nine teams with only the math task and five teams with only the physics task. 18 of the 74 sessions occurred in the lab, and the remaining 56 sessions occurred in school environments. The dataset consists of 8,660 utterances coded with CPS skills, and corresponding transcripts. Of these utterances, 2,751 (32%) were from lab sessions and the other 5,909 (68%) were from school sessions.

2.5 Machine Learning

We adopted a supervised classification approach to predict the ground truth CPS skill for each utterance. We first implemented a bag-of-n-grams approach using a Random Forest Classifier, as recent literature [65] has shown this method to be effective for the classification of CPS utterances. Next, we explored deep transfer learning as a means to improve upon this method. In particular, we leveraged pre-trained language models and employed the popular Bidirectional Encoder Representations from Transformers (BERT) model [18]. Additionally, we tested a method (BERT-seq) which takes a sequence of utterances as input (the utterance to classify plus the previous and subsequent utterances) to capture contextual information, in order to determine if including adjacent utterances improves classification accuracy. We trained separate models (RF, BERT, and BERT-seq) using the ASR transcripts and human transcripts as input.

2.5.1 Random Forest N-Grams

We first followed the approach outlined in [65] and trained Random Forest Classifiers to predict the CPS skill for each utterance using n-gram features. We used unigrams (words) and bigrams (two-word phrases) as the features for our Random Forest classifiers. Trigrams and beyond were not used since very few unique trigrams (only 6) occurred in >1% of utterances. We explored excluding n-grams that occurred at less than a minimum frequency in the training dataset, testing values of 0% (no filtering), 1% and 2% as hyperparameters. We used the scikit-learn [52] library's implementation of the Random Forest Classifier with 200 estimators.

2.5.2 BERT

We used a transfer learning approach and fine-tuned pre-trained BERT models to predict the CPS skill for each utterance. This entailed starting with a BERT model pre-trained on a large amount of unlabeled data, then fine-tuning it on our dataset of transcribed utterances and corresponding labels (CPS skills). We first processed the transcribed utterances using WordPiece tokenization [61]. This process entailed splitting an utterance into a sequence of words, or parts of words. Each unique word or word piece was then converted to an integer (called a token) according to BERT's pre-specified vocabulary. Finally, special tokens ([CLS] and [SEP]) were appended to the beginning and end of this sequence of integers and the sequence was provided as input to BERT (see Figure 2A). BERT mapped each input token to a 768-dimensional embedding, which serves as a semantic representation of the input token (the embedding of the special [CLS] and [SEP] tokens capture a semantic representation of the entire sequence of input tokens).

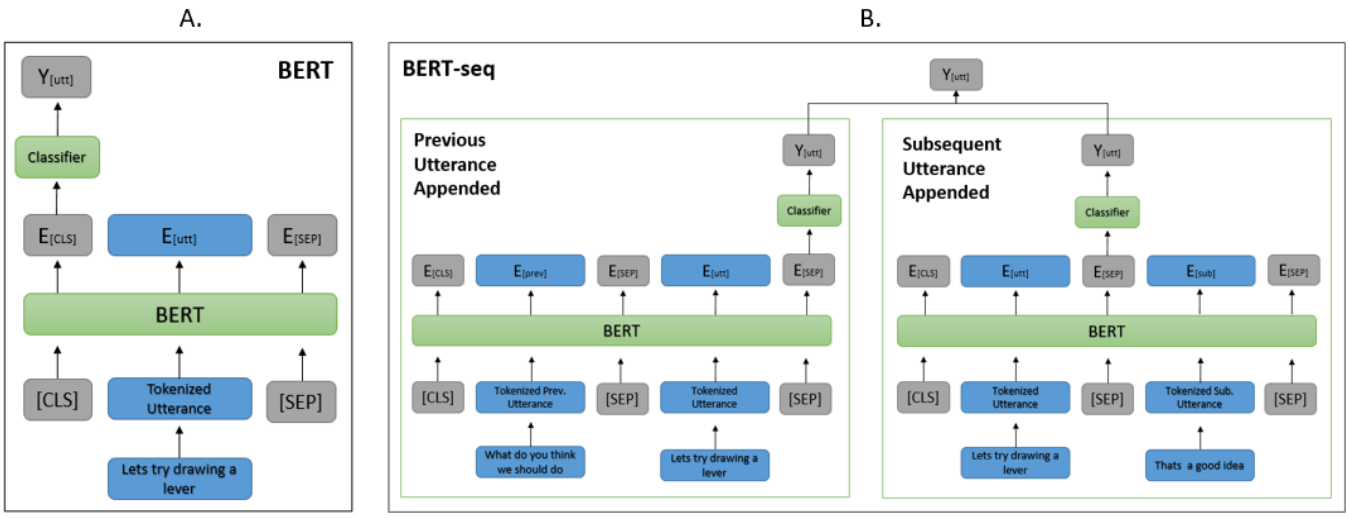


Figure 2. (A) The traditional BERT model used for text classification. (B) Our BERT-seq model which captures contextual information from the previous and subsequent utterances during classification.

For classification, the embedding of the [CLS] token was used as input to a fully connected layer (classifier), which output predicted probabilities for the seven CPS skills. We used multiclass learning, meaning that all seven CPS skills were predicted by one model.

2.5.3 BERT-seq

We propose a method to incorporate contextual utterances during classification by creating a special input representation, without augmenting the BERT architecture. This method takes a sequence of three utterances as input (the utterance to classify plus the previous and subsequent utterances), which are used to train two separate BERT models, each including either the previous or subsequent utterance in the BERT input (see Figure 2B). To add a pair of adjacent utterances to the input, we first processed each utterance individually using WordPiece tokenization as described above. The special [CLS] token was then added to the beginning of this sequence, and a [SEP] token was added to the end of both the first and second utterances. To classify the utterance, the embedding of the corresponding [SEP] token was used as input to a fully connected layer, which output predictions for the 7 CPS skills. Finally, the predicted probabilities of the previous and subsequent utterance models were averaged. This method of representing a sequence of utterances enables the self-attention layers of BERT to leverage contextual information from the previous and subsequent utterances, while still utilizing the pre-trained BERT weights.

For both BERT and BERT-seq we started with the transformers [73] library's implementation of the BertModel with the "bert-base-uncased" pre-trained weights, and used the BertTokenizer to process our utterances. We then fine-tuned the models for three epochs using a batch size of 16. We found that fine-tuning beyond three epochs did not substantially improve model performance.

2.5.4 Cross Validation

We used team-level 10-fold cross-validation to assess the accuracy of our classifiers. With our dataset of 44 teams, this entailed training a model with utterances from 90% of teams (39 or 40 teams), then evaluating the model's predictive accuracy on a test set containing utterances from the 10% of teams withheld during training (4 or 5 teams). This process was repeated ten times, such that every team appeared in the test set once. To compute accuracy metrics, predictions from all ten folds were aggregated and a single metric was computed on the full dataset. Team-level cross validation yields a better assessment of the method's generalizability to new teams because it ensures each model is never trained and evaluated on utterances from the same speaker. We used identical cross-validation folds for the RF, BERT and BERT-seq models as well as the human and ASR transcripts to ensure that differences in performance were not an artifact of the folds used. This experiment was repeated for 5 iterations, and different randomized cross-validation folds were used for each iteration.

3. RESULTS

3.1 ASR Accuracy

We compared WER in the lab and school subsets in order to quantify the speech recognition error that could be attributed to noisy school environments, as opposed to other factors such as difficulty recognizing children's speech, whispering or mumbling, audio quality, or inevitable ASR mistakes. We used the Human Transcript Subset as described in Section 2.3 for this comparison. The distributions of WER in the lab and school environments are shown in Figure 3. We found that WER was much lower in the lab environment than in schools (mean WER of .54 and .76, median WER of .50 and .91, respectively), indicating that significant ASR error is due to noisy school environments. We performed a non-parametric Kruskal-Wallis test [40] to statistically compare WER in the lab and school samples, and found that they differed significantly ($\chi^2(1) = 62.13$, $p < .001$).

As evident in Figure 3, a large proportion (47%) of the school utterances had a WER of 1 (compared to 19% for lab data), meaning no words were correctly recognized. However, WER was also high in the controlled lab environment, suggesting that speech recognition error may in part be attributable to factors beyond the complications of noisy school environments.

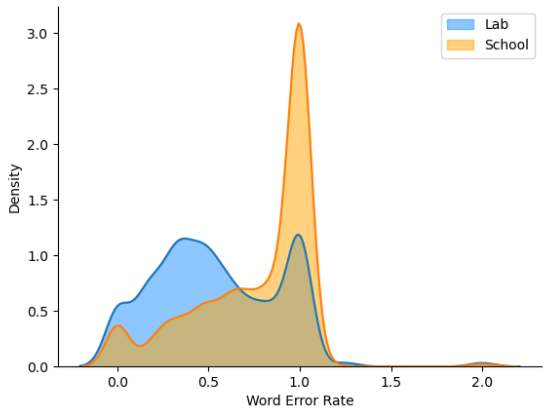


Figure 3. Gaussian kernel density estimates of the distribution of word error rates in the lab and school environments.

We also investigated the correlation between WER and ASR confidence to determine whether the confidence values produced by the ASR provided a good estimate of transcript accuracy. We found that WER and ASR confidence were significantly correlated (Spearman *rho* = -.74, $p < .001$).

3.2 Model Comparison

Next we compared the performance of our three NLP models (RF, BERT, BERT-seq). The models output a probability from 0 to 1 that an utterance is coded with each CPS skill. Accordingly, we report the area under the receiver operating characteristic curve (AUROC) for each skill, a common accuracy metric for model performance [6] which takes into account the true positive and false positive tradeoff across classification thresholds. Mean AUROC scores (over the five iterations) for the RF, BERT and BERT-seq models, using both human and ASR transcripts are reported in Table 2. We also report a chance baseline, created by randomly shuffling the labels within each CPS session and computing accuracy accordingly. Because shuffling is within sessions, the AUROCs for the shuffled models will slightly deviate from the 0.5 chance baseline. To determine if the three model's AUROC scores were significantly different for each CPS skill, we used a bootstrap method to statistically compare the AUROC values. Since five iterations of this experiment were conducted, we selected the model corresponding to the median AUROC value across the five iterations (for both human and ASR transcripts) on each CPS skill for statistical analysis. We performed this analysis in R using the pROC package [56] with 2,000 bootstrap permutations. Finally, we adjusted the resulting p-values using a false discovery rate (FDR) correction [5] to account for multiple testing across the seven CPS skills.

Without exception BERT-seq quantitatively yielded the highest AUROC scores for all seven CPS skills using both human and ASR transcripts, indicating that our method of incorporating adjacent utterances improves performance over single utterance classifiers. On average, BERT outperformed the RF model on both human and ASR transcripts, although there were some skills for which the RF AUROC scores were higher. From the statistical analysis described above, we found that with ASR transcripts BERT-seq had a significant advantage over the other two models for most skills (four of seven for BERT, five of seven for RF). We also found that there was no significant difference between BERT and RF for six of seven skills.

Table 2. Mean AUROC values (across 5 iterations) of the RF N-gram, BERT, and BERT-seq models on ASR and Human transcripts for all CPS skills.

CPS Skill	ASR Transcripts			Human Transcripts			
	RF	**BERT**	**BERT-seq**	**RF**	**BERT**	**BERT-seq**	**Shuffled**
Sharing Information	0.711	0.745 ^{R}	0.756 ^{R}	0.837	0.866 ^{R}	0.877 ^{R}	0.540
Establishing Shared Understanding	0.713	0.724	0.740 ^{RB}	0.872	0.894 ^{R}	0.907 ^{RB}	0.509
Negotiating	0.721	0.719	0.741 ^{B}	0.896	0.901	0.916 ^{RB}	0.510
Executing	0.745	0.767	0.784 ^{R}	0.897	0.914 ^{R}	0.926 ^{R}	0.574
Maintaining Communication	0.673	0.667	0.750 ^{RB}	0.849	0.853	0.901 ^{RB}	0.557
Monitoring	0.632	0.594	0.677 ^{RB}	0.812	0.792	0.843 ^{RB}	0.513
Planning	0.700	0.692	0.718	0.861 ^{B}	0.818	0.872 ^{B}	0.502
Micro Avg.	0.773	0.782	0.799	0.887	0.895	0.914	0.607

^{R} and ^{B} indicate the AUROC score was significantly higher than the RF and/or BERT models, respectively. Neither RF nor BERT ever outperformed BERT-seq.

We observed a similar pattern on the human transcripts, where BERT-seq significantly outperformed BERT on five of seven skills and RF on six of seven skills. Interestingly, on human transcripts the advantage of BERT over RF increased, with BERT having significantly higher scores on three skills, while RF was significantly better on only one. This finding suggests that with high quality transcripts which accurately capture the content of an utterance, BERT was the better model, whereas with noisy ASR transcripts there was no clear difference.

These results indicate that BERT-seq quantitatively outperformed both the traditional BERT and the RF n-gram approach for all seven CPS skills, using both the human and ASR transcripts. However, the statistical analysis revealed that for some CPS skills, this advantage was not statistically significant. As BERT-seq was the best model across CPS skills, we refer to these results in our comparison of human and ASR transcripts, and throughout the rest of this paper.

3.3 ASR vs. Human Transcripts

We found that using the ASR transcripts as input, our best model (BERT-seq) was able to accurately classify the seven CPS skills, yielding a micro-average AUROC score of .799. However, when the human transcripts were used, this average increased to .914 (see Table 2). We compared the human and ASR transcript results using the bootstrap method described above, and found that the human transcript AUROC scores were significantly (FDR corrected $p < .05$) higher than the ASR transcript scores for all seven CPS skills, an unsurprising result given the high word error rates in the ASR transcripts. However, we note that despite significant loss in performance due to speech recognition error, our model easily outperformed a shuffled baseline (micro-average AUROC of .607), supporting the hypothesis that CPS skills can be automatically predicted from ASR transcripts.

3.4 Classification Accuracy in Lab and School Environments

Next we compared classification accuracy in the lab and school environments in order to investigate the extent to which higher rates of ASR error in the school subset affected model performance. We report AUROC scores for the lab and school environments in Table 3. We found that on average, classification accuracy was substantially lower in the school subset compared to the lab subset (micro-average AUROC of .783 and .830, respectively). Further, for every individual skill, AUROC scores were quantitatively higher in the lab subset than in the school subset, with differences in AUROC values for individual skills ranging from .031 (Executing) to .102 (Negotiating). We again used the bootstrap method to statistically compare AUROC scores in the lab and school for each skill and found that scores were significantly higher in the lab subset for five out of seven CPS skills (see Table 3).

3.5 Classification Accuracy as a Function of ASR Confidence

Lastly, we examined the relationship between ASR confidence and classification accuracy. As discussed in section 3.1, the ASR confidence is a good proxy for word error rate, as the two values are significantly correlated. Therefore, we separated our 8,660 utterances into ten ASR confidence bins (0.0 – 0.1, etc.) and computed the micro-average AUROC score for each bin. The distribution of utterances and corresponding AUROC scores for each bin are shown in Figure 4A and 4B, respectively. Figure 4B also shows the human transcript AUROC score as a benchmark of the accuracy that would be expected under conditions of near-perfect speech recognition. The shuffled baseline is also shown to visualize improvement over chance.

Table 3. Mean AUROC scores (across 5 iterations) for each CPS skill in Lab and School environments. Results are from the BERT-seq model using ASR transcripts. Values marked with * were significantly higher in the Lab vs. School.

CPS Skill	Lab		School	
	AUC	**Base Rate**	**AUC**	**Base Rate**
Sharing Information	0.782*	.25	0.743	.27
Establishing Shared Understanding	0.786*	.26	0.716	.25
Negotiating	0.807*	.18	0.705	.15
Executing	0.804	.15	0.773	.13
Maintaining Communication	0.803*	.03	0.717	.08
Monitoring	0.701	.05	0.663	.07
Planning	0.760*	.06	0.688	.04
Micro Avg.	0.830		0.783	

We found that a large proportion of utterances (20%) fall in the [0.0 - 0.1) bin, indicating that the ASR had little to no confidence in their content. In fact, nearly all (97%) of the utterances in this bin have an empty ASR transcript, meaning no words were recognized during the utterance's segmented time window. In many cases, this occurred due to the students whispering or mumbling, which the ASR was unable to recognize. Excepting the significant zero inflation, the utterances appeared to be normally distributed around the [0.6 - 0.7) bin.

We observed a strong correlation between ASR confidence bin and classification accuracy (Spearman $rho = .94$, $p < .001$). Unsurprisingly, we found that for low confidence transcripts (< 0.3) a substantial gap exists between the ASR transcript AUROC score and the benchmark human transcript score (see Figure 4B). On these low confidence transcripts, model performance is near the shuffled chance baseline. Interestingly, despite many (77%) of these low confidence transcripts containing no words, the model was still able to outperform the chance baseline by learning the distribution of skills among empty transcripts in the training data. We found that accuracy increases steadily among the medium confidence transcripts (0.3 - 0.7). For high confidence transcripts (≥0.7), AUROC scores are near (though still lower than) the benchmark human transcript values. The

relationship between ASR confidence and classification accuracy indicates that it might be viable to filter out utterances with low confidence to improve reliability for downstream applications.

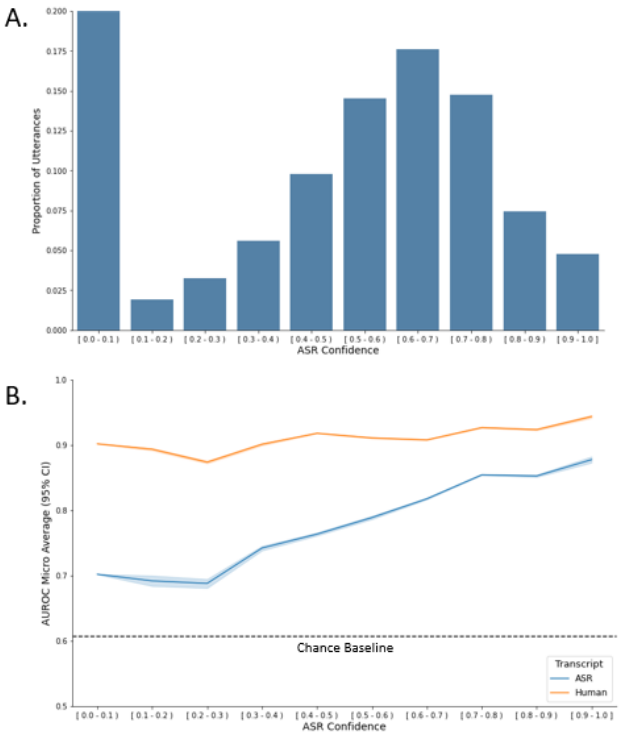


Figure 4. (A) Distribution of ASR confidence on all 8,660 utterances. (B) Model accuracy as a function of ASR confidence. Micro-average AUROC scores across the 7 CPS skills (with 95% CI across 5 iterations) are plotted for Human and ASR transcripts.

4. DISCUSSION

We investigated the feasibility of using automatic speech recognition and natural language processing to automatically classify student speech with CPS skills using data collected in both lab and real-world school environments. We compared performance using imperfect ASR transcripts with human transcripts, investigated differences between the lab and school environments, and explored three NLP approaches including bag-of-n-grams and deep transfer learning. In the rest of this section we discuss our main findings, applications of our models, as well as limitations and future directions of research.

4.1 Main Findings

We found that it is feasible to use ASR to transcribe middle and high school student's speech during CPS in both lab and school environments. However, we found that significant speech recognition error is introduced when speech is recorded in schools (mean WER of .76), likely as a result of noisy environments and distractions from other students. That said, speech recognition error was also high in the lab environment (mean WER of .54), suggesting that there may still be fundamental limitations associated with using ASR on children's speech in the context of remote CPS.

Despite imperfect speech recognition, we demonstrated that it is possible to automatically predict CPS skills from student speech in a real-world school environment. We built team-independent models that were able to predict CPS skills with reasonable accuracy (micro-average AUROC of .80) using ASR transcripts. Importantly, this result outperformed a shuffled baseline (micro-average AUROC of .61) by a significant margin. This finding is encouraging because it was previously unknown whether ASR could yield transcripts of sufficient quality to model CPS skills in noisy environments. Further, we demonstrated that by using high-fidelity human transcripts, this accuracy could be significantly improved (micro-average AUROC of .91). We demonstrated that in the absence of ASR error our NLP models were highly accurate, suggesting a useful upper bound of what can be achieved from spoken content alone.

We also improved upon NLP approaches previously used in CPS literature, demonstrating the advantage of deep transfer learning over standard classifiers for modeling CPS language. We found that on average, using both ASR and human transcripts, the deep transfer learning model (BERT) achieved slightly better accuracy than the Random Forest n-gram model (though the two were statistically tied for 3/7 CPS skills with human transcripts and 6/7 skills with ASR transcripts). This finding was unsurprising given that pre-trained language models have achieved state-of-the-art performance on many NLP benchmark tasks, including text classification.

Importantly, we found that we were able to further improve classification accuracy by constructing an input representation that enables BERT to capture information from adjacent utterances. This method showed significant improvement over the single utterance BERT and RF models, providing preliminary evidence of its viability. This finding suggests that in CPS, the context of an utterance (what was said before and after) may be important for accurate identification of particular CPS skills.

Finally, we examined the relationship between ASR confidence – a proxy for transcription quality – and classification accuracy. We found that the two were highly correlated, suggesting that downstream applications may be able to improve reliability of predictions by filtering out low confidence transcripts.

4.2 Applications

A key application of this work is the automatic assessment of CPS skills from open-ended speech in classrooms and beyond. As previously discussed, analyzing verbal communication for evidence of CPS skills is a costly and time-intensive process when trained human coders are used. Our findings suggest that automated methods using ASR and NLP may provide a viable alternative to the human-coding process. These automated methods hold great potential in improving the assessment and training of CPS skills, a priority of modern education [49]. However, given the imperfect accuracy of our models, and unanswered questions regarding how this approach may generalize to students with differing communication styles or cultural and linguistic backgrounds, this approach should be limited to formative assessment [63] focused on learning and improvement, rather than evaluation.

Our approach could advance this goal in several ways. For example, automatically generated reports could be sent to a teacher monitoring many groups of students engaged in CPS, informing the teacher of the extent to which each group is demonstrating CPS skills. Such a system could help the teacher

identify which groups need support and allocate their limited presence toward assisting those groups. Similarly, these reports could be used to identify individual student's strengths and weaknesses, and set appropriate goals for improvement. For instance, a student who frequently shares information yet seldom engages in negotiation or establishing shared understanding could be encouraged to listen to the ideas of their teammates and work to build on those ideas together.

In addition to passive assessment and off-line feedback, this approach could be leveraged by next-generation intelligent systems that actively monitor ongoing CPS and dynamically intervene in real time to yield improved CPS outcomes [15], or provide personalized on-line feedback to students. For example, a group frequently engaging in off-topic conversation could be prompted by the system to focus back on the problem-solving task, or a particular student within a group who hasn't shared information could be encouraged to share their ideas with the team. The specific intervention strategies, including when to intervene, how to present the intervention, and who the intervention should be targeted at (whole group vs. individual student) await design, testing, and refinement.

Importantly, a technology devised to assist in the training and assessment of CPS does little good if it is confined to the lab. Thus, the present results take a step towards the development of a system that can support CPS in real-world classrooms by monitoring open-ended verbal communication for CPS skills.

4.3 Limitations

There were some limitations of this work. First, although we used an automated approach for utterance transcription and CPS skill prediction, the sessions were segmented into utterances beforehand by human coders. This is a limitation because a fully automated pipeline would require the ASR to automatically detect and segment recorded speech into individual utterances, an already difficult task that may be further complicated by noisy school environments or the peculiarities of children's speech. Another related limitation is that due to the utterance segmentation and ASR transcription process we used, our ASR transcripts contain all speech that was recognized during an utterance's segmented time window. This means that some ASR transcripts contain words from both speakers, which introduces alignment inconsistencies between the ASR transcript and the coded CPS skill because utterances were coded at the individual student level. In particular, this introduces noise into the ASR transcripts when student's utterances overlap.

Another limitation of this work is that we considered only linguistic features to predict the coded CPS skills. We expect that model performance can be improved by modeling not only what students say (language), but considering how they say it (acoustic-prosodic information) and in the context of what they're doing (task-specific information). We hypothesize that the inclusion of these additional modalities may particularly improve performance for low confidence ASR transcripts, where the language transcribed by the speech recognizer is either missing altogether, or is a poor representation of what was actually said. Finally, although we demonstrated that our method for capturing contextual information from adjacent utterances improved accuracy, we did not compare this with other methods for incorporating contextual utterances such as conditional random fields or recurrent neural networks.

4.4 Future Work

The findings and limitations discussed in this section present several possibilities for improvement in future research. First, in order to develop a fully automated approach for modeling CPS skills, we plan to incorporate automatic utterance segmentation and speaker diarization into our ASR pipeline. Further, we plan to explore methods for incorporating information from other modalities in addition to language. For instance, including features such as acoustic-prosodic information, task context, facial expression, or body movement may enable more accurate prediction of CPS skills in cases where ASR fails to capture the content of an utterance.

Another direction of future research involves further exploration of how contextual utterances can be used to improve classification accuracy. We demonstrated a method for incorporating adjacent utterances in our model input, which improved performance over single utterance classifiers. In future work, we will explore methods for capturing contextual information beyond the previous and subsequent utterances (e.g., the five previous utterances). We also plan to investigate how the approach demonstrated in this paper, which leverages the model's attention mechanism to capture context, compares with other approaches (e.g., recurrent neural networks).

In addition to exploring methods for improving the accuracy of our models, we plan to investigate the utility of our CPS models. An open question is how accurate model predictions need to be to provide useful and actionable estimates for assessment, feedback, or intervention. Specifically, recent work [2, 25] has clustered students using the frequency of CPS skills to derive theoretically grounded profiles of collaborative problem solvers (e.g., active collaborators, social loafers). We plan to investigate whether model-derived estimates of CPS skill frequencies will yield high agreement to the clustering produced using human codes.

5. CONCLUSION

We combined automatic speech recognition and natural language processing to automatically predict CPS skills from student speech during problem solving in both lab and real-world school environments. Our findings suggest that despite significant speech recognition error in school environments, it is possible to predict expert-coded CPS skills using automatically generated transcripts. These findings open many possibilities for next-generation technologies that can further the goal of improved CPS training, assessment, and support in schools.

6. ACKNOWLEDGMENTS

This research was supported by the Institute of Educational Sciences (IES R305A170432), the NSF National AI Institute for Student-AI Teaming (iSAT) (DRL 2019805) and NSF DUE 1745442/1660877. The opinions expressed are those of the authors and do not represent views of the funding agencies.

7. REFERENCES

[1] Andrews-Todd, J. et al. 2019. Collaborative Problem Solving Assessment in an Online Mathematics Task. *ETS Research Report Series*. 2019, 1 (2019). DOI:https://doi.org/10.1002/ets2.12260.

[2] Andrews-Todd, J. et al. 2018. Identifying profiles of collaborative problem solvers in an online electronics environment. *Proceedings of the 11th International Conference on Educational Data Mining, EDM 2018* (2018).

[3] Andrews-Todd, J. and Forsyth, C.M. 2020. Exploring social and cognitive dimensions of collaborative problem solving in an open online simulation-based task. *Computers in Human Behavior*. 104, (2020). DOI:https://doi.org/10.1016/j.chb.2018.10.025.

[4] Andrews-Todd, J. and Kerr, D. 2019. Application of Ontologies for Assessing Collaborative Problem Solving Skills. *International Journal of Testing*. 19, 2 (2019). DOI:https://doi.org/10.1080/15305058.2019.1573823.

[5] Benjamini, Y. and Hochberg, Y. 1995. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B (Methodological)*. 57, 1 (1995). DOI:https://doi.org/10.1111/j.2517-6161.1995.tb02031.x.

[6] Bradley, A.P. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*. 30, 7 (1997). DOI:https://doi.org/10.1016/S0031-3203(96)00142-2.

[7] C. Graesser, A. et al. 2018. Challenges of Assessing Collaborative Problem Solving. *Care, E., Griffin, P., Wilson, M. (Eds.), Assessment and teaching of 21st century skills: Research and applications*. 75–91.

[8] Care, E. et al. 2016. Assessment of Collaborative Problem Solving in Education Environments. *Applied Measurement in Education*. 29, 4 (2016). DOI:https://doi.org/10.1080/08957347.2016.1209204.

[9] Chen, S. et al. 1998. Evaluation metrics for language models. *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*. (1998).

[10] Chopade, P. et al. 2019. CPSX: Using AI-Machine Learning for Mapping Human-Human Interaction and Measurement of CPS Teamwork Skills. *2019 IEEE International Symposium on Technologies for Homeland Security, HST 2019* (2019).

[11] Cicchetti, D. V. 1994. Guidelines, Criteria, and Rules of Thumb for Evaluating Normed and Standardized Assessment Instruments in Psychology. *Psychological Assessment*. 6, 4 (1994). DOI:https://doi.org/10.1037/1040-3590.6.4.284.

[12] Cohan, A. et al. 2020. Pretrained language models for sequential sentence classification. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference* (2020).

[13] Cukurova, M. et al. 2020. Modelling collaborative problem-solving competence with transparent learning analytics: Is video data enough? *ACM International Conference Proceeding Series* (2020).

[14] Cukurova, M. et al. 2018. The NISPI framework: Analysing collaborative problem-solving from students' physical interactions. *Computers and Education*. 116, (2018). DOI:https://doi.org/10.1016/j.compedu.2017.08.007.

[15] D'Mello, S. et al. 2019. Towards dynamic intelligent support for collaborative problem solving. *CEUR Workshop Proceedings* (2019).

[16] von Davier, A.A. et al. 2017. Interdisciplinary research agenda in support of assessment of collaborative problem solving: lessons learned from developing a Collaborative Science Assessment Prototype. *Computers in Human Behavior*. 76, (2017). DOI:https://doi.org/10.1016/j.chb.2017.04.059.

[17] Dedoose version 8.0.35 2018. Dedoose: Web applicaiton for managing, analyzing, and presenting qualitative and mixed method research data. *SocioCultural Research Consultants, LLC*.

[18] Devlin, J. et al. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* (2019).

[19] Dowell, N.M.M. et al. 2020. Exploring the relationship between emergent sociocognitive roles, collaborative problem-solving skills, and outcomes: A group communication analysis. *Journal of Learning Analytics*. 7, 1 (2020). DOI:https://doi.org/10.18608/jla.2020.71.4.

[20] Dowell, N.M.M. et al. 2019. Group communication analysis: A computational linguistics approach for detecting sociocognitive roles in multiparty interactions. *Behavior Research Methods*. 51, 3 (2019). DOI:https://doi.org/10.3758/s13428-018-1102-z.

[21] Emara, M. et al. 2021. Examining Student Regulation of Collaborative, Computational, Problem-Solving Processes in Open-Ended Learning Environments. *Journal of Learning Analytics*. 8, 1 (2021). DOI:https://doi.org/10.18608/jla.2021.7230.

[22] Felstead, A. and Henseke, G. 2017. Assessing the growth of remote working and its consequences for effort, well-being and work-life balance. *New Technology, Work and Employment*. 32, 3 (2017). DOI:https://doi.org/10.1111/ntwe.12097.

[23] Fiore, S.M. et al. 2018. Collaborative problem-solving education for the twenty-first-century workforce. *Nature Human Behaviour*.

[24] Flor, M. et al. 2016. Automated classification of collaborative problem solving interactions in simulated science tasks. (2016).

[25] Forsyth, C. et al. 2020. Are You Really A Team Player ? Profiling of Collaborative Problem Solvers in an Online

Environment. *Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020*. Edm (2020).

[26] Gerosa, M. et al. 2009. A review of ASR technologies for children's speech. *Proceedings of the 2nd Workshop on Child, Computer and Interaction, WOCCI '09* (2009).

[27] Graesser, A.C. et al. 2018. Advancing the Science of Collaborative Problem Solving. *Psychological Science in the Public Interest*. 19, 2 (2018), 59–92. DOI:https://doi.org/10.1177/1529100618808244.

[28] Griffin, P. et al. 2012. The changing role of education and schools. *Assessment and teaching of 21st century skills*.

[29] Hao, J. et al. 2017. CPS-Rater: Automated Sequential Annotation for Conversations in Collaborative Problem-Solving Activities. *ETS Research Report Series*. 2017, 1 (2017). DOI:https://doi.org/10.1002/ets2.12184.

[30] Hesse, F. et al. 2015. A Framework for Teachable Collaborative Problem Solving Skills. *Assessment and Teaching of 21st Century Skills*.

[31] Howard, J. and Ruder, S. 2018. Universal language model fine-tuning for text classification. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)* (2018).

[32] Huang, K. et al. 2019. Identifying collaborative learning states using unsupervised machine learning on eye-tracking, physiological and motion sensor data. *EDM 2019 - Proceedings of the 12th International Conference on Educational Data Mining* (2019).

[33] IBM Watson: *https://www.ibm.com/watson/services/speech-to-text/*. Accessed: 2021-03-02.

[34] Kerr, D. et al. 2016. The In-Task Assessment Framework for Behavioral Data. *The Handbook of Cognition and Assessment*.

[35] Kim, Y.J. and Shute, V.J. 2015. Opportunities and Challenges in Assessing and Supporting Creativity in Video Games. *Video Games and Creativity*.

[36] Kniffin, K.M. et al. 2020. COVID-19 and the Workplace: Implications, Issues, and Insights for Future Research and Action. *American Psychologist*. (2020). DOI:https://doi.org/10.1037/amp0000716.

[37] Koenig, J.A. 2011. *Assessing 21st Century Skills: Summary of a Workshop*.

[38] Lai, E. et al. 2017. *Skills for today: What We Know about Teaching and Assessing Collaboration*.

[39] Liu, L. et al. 2015. A tough nut to crack: Measuring collaborative problem solving. *Handbook of Research on Technology Tools for Real-World Skill Development*.

[40] McKight, P.E. and Najab, J. 2010. Kruskal-Wallis Test. *The Corsini Encyclopedia of Psychology*.

[41] Mislevy, R.J. et al. 2003. Focus Article: On the Structure of Educational Assessments. *Measurement: Interdisciplinary Research & Perspective*. 1, 1 (2003). DOI:https://doi.org/10.1207/s15366359mea0101_02.

[42] Miura, G. and Okada, S. 2019. Task-independent multimodal prediction of group performance based on product dimensions. *ICMI 2019 - Proceedings of the 2019 International Conference on Multimodal Interaction* (2019).

[43] Müller, P. et al. 2018. Detecting low rapport during natural interactions in small groups from non-verbal behaviour. *International Conference on Intelligent User Interfaces, Proceedings IUI* (2018).

[44] Murray, G. and Oertel, C. 2018. Predicting group performance in task-based interaction. *ICMI 2018 - Proceedings of the 2018 International Conference on Multimodal Interaction* (2018).

[45] Narayanan, S. and Potamianos, A. 2002. Creating conversational interfaces for children. *IEEE Transactions on Speech and Audio Processing*. 10, 2 (2002). DOI:https://doi.org/10.1109/89.985544.

[46] O'Neil, H.F.. C.G.K.W.K.. B.R.S. 1995. *Measurement of teamwork processes using computer simulation (CSE Tech. Rep. No. 399).*

[47] O'neil, H.F. et al. 2010. Computer-based feedback for computer-based collaborative problem solving. *Computer-Based Diagnostics and Systematic Analysis of Knowledge.*

[48] OECD 2013. *PISA 2012 Assessment and Analytical Framework: Mathematics, reading, science, problem solving and financial literacy*.

[49] OECD 2015. Pisa 2015 Collaborative Problem Solving Framework. (2015).

[50] Olsen, J.K. et al. 2020. Temporal analysis of multimodal data to predict collaborative learning outcomes. *British Journal of Educational Technology*. 51, 5 (2020). DOI:https://doi.org/10.1111/bjet.12982.

[51] Oviatt, S. and Cohen, A. 2013. Written and multimodal representations as predictors of expertise and problem-solving success in mathematics. *ICMI 2013 - Proceedings of the 2013 ACM International Conference on Multimodal Interaction* (2013).

[52] Pedregosa, F. et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*. 12, (2011).

[53] Potamianos, A. and Narayanan, S. 2003. Robust Recognition of Children's Speech. *IEEE Transactions on Speech and Audio Processing*. 11, 6 (2003). DOI:https://doi.org/10.1109/TSA.2003.818026.

[54] Prata, D.N. et al. 2009. Detecting and understanding the impact of cognitive and interpersonal conflict in computer supported collaborative learning environments. *EDM'09 - Educational Data Mining 2009: 2nd International Conference on Educational Data Mining* (2009).

[55] Reilly, J.M. and Schneider, B. 2019. Predicting the

quality of collaborative problem solving through linguistic analysis of discourse. *EDM 2019 - Proceedings of the 12th International Conference on Educational Data Mining* (2019).

[56] Robin, X. et al. 2011. pROC: An open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*. 12, (2011). DOI:https://doi.org/10.1186/1471-2105-12-77.

[57] Roschelle, J. and Teasley, S.D. 1995. The Construction of Shared Knowledge in Collaborative Problem Solving. *Computer Supported Collaborative Learning*.

[58] Rosé, C. et al. 2008. Analyzing collaborative learning processes automatically: Exploiting the advances of computational linguistics in computer-supported collaborative learning. *International Journal of Computer-Supported Collaborative Learning*. 3, 3 (2008). DOI:https://doi.org/10.1007/s11412-007-9034-0.

[59] Samrose, S. et al. 2018. CoCo: Collaboration Coach for Understanding Team Dynamics during Video Conferencing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 1, 4 (2018). DOI:https://doi.org/10.1145/3161186.

[60] Schulze, J. and Krumm, S. 2017. The "virtual team player": A review and initial model of knowledge, skills, abilities, and other characteristics for virtual collaboration. *Organizational Psychology Review*. 7, 1 (2017). DOI:https://doi.org/10.1177/2041386616675522.

[61] Schuster, M. and Nakajima, K. 2012. Japanese and Korean voice search. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (2012).

[62] Shute, V.J. et al. 2013. Assessment and learning of qualitative physics in Newton's playground. *Journal of Educational Research*. 106, 6 (2013). DOI:https://doi.org/10.1080/00220671.2013.832970.

[63] Shute, V.J. 2008. Focus on formative feedback. *Review of Educational Research*. 78, 1 (2008). DOI:https://doi.org/10.3102/0034654307313795.

[64] Spada, H. et al. 2005. A new method to assess the quality of collaborative process in CSCL. *Computer Supported Collaborative Learning 2005: The Next 10 Years - Proceedings of the International Conference on Computer Supported Collaborative Learning 2005, CSCL 2005* (2005).

[65] Stewart, A.E.B. et al. 2019. I say, you say, we say: Using spoken language to model socio-cognitive processes during computer-supported collaborative problem solving. *Proceedings of the ACM on Human-Computer Interaction*. 3, CSCW (2019). DOI:https://doi.org/10.1145/3359296.

[66] Stewart, A.E.B. et al. 2021. Multimodal modeling of collaborative problem-solving facets in triads. *User Modeling and User-Adapted Interaction*. (2021). DOI:https://doi.org/10.1007/s11257-021-09290-y.

[67] Subburaj, S.K. et al. 2020. Multimodal, Multiparty Modeling of Collaborative Problem Solving Performance. *ICMI 2020 - Proceedings of the 2020 International Conference on Multimodal Interaction* (2020).

[68] Sun, C. et al. 2020. Towards a generalized competency model of collaborative problem solving. *Computers and Education*. 143, (2020). DOI:https://doi.org/10.1016/j.compedu.2019.103672.

[69] Suresh, A. et al. 2021. Using Transformers to Provide Teachers with Personalized Feedback on their Classroom Discourse: The TalkMoves Application. *AAAI Spring Symposium Series 2021*.

[70] Vaessen N 2019. Word error rate for automatic speech recognition. https: //pypi.org/project/jiwer/.

[71] Vaswani, A. et al. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[72] Vrzakova, H. et al. 2020. Focused or stuck together: Multimodal patterns reveal triads' performance in collaborative problem solving. *ACM International Conference Proceeding Series* (2020).

[73] Wolf, T. et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv*.

Just a Few Expert Constraints Can Help: Humanizing Data-Driven Subgoal Detection for Novice Programming

Samiha Marwan, Yang Shi, Ian Menezes, Min Chi, Tiffany Barnes, Thomas W. Price
North Carolina State University, Raleigh, NC, USA
samarwan, yshi26, ivmeneze, mchi, tmbarnes, twprice@ncsu.edu

ABSTRACT

Feedback on how students progress through completing subgoals can improve students' learning and motivation in programming. Detecting subgoal completion is a challenging task, and most learning environments do so either with *expert-authored* models or with *data-driven* models. Both models have advantages that are complementary – expert models encode domain knowledge and achieve reliable detection but require *extensive authoring efforts* and often cannot capture all students' possible solution strategies, while data-driven models can be easily scaled but may be less accurate and interpretable. In this paper, we take a step towards achieving the best of both worlds – utilizing a data-driven model that can intelligently detect subgoals in students' correct solutions, while benefiting from human expertise in editing these data-driven subgoal rules to provide more accurate feedback to students. We compared our hybrid "humanized" subgoal detectors, built from data-driven subgoals modified with expert input, against an existing data-driven approach and baseline supervised learning models. Our results showed that the hybrid model outperformed all other models in terms of overall accuracy and F1-score. Our work advances the challenging task of automated subgoal detection during programming, while laying the groundwork for future hybrid expert-authored/data-driven systems.

Keywords

Subgoals, Formative feedback, Data-driven hybrid models

1. INTRODUCTION

Formative feedback has been shown to be an effective form of automated feedback that can improve students' learning and motivation [54, 38, 8, 20]. In programming, *immediate* formative feedback *during* problem-solving is important because some problems require students to find one of many correct solutions [16], and novices may be uncertain about when they are on the right track [62]. This uncertainty may lead some students to give up [43], and can also negatively impact student confidence and motivation in computer science (CS) [32]. Prior research has shown that immediate feedback can address this, reducing novices' uncertainty and improving their confidence, engagement, motivation, and learning [42, 8, 33, 21, 38, 39].

One effective form of immediate feedback is subgoal feedback [40], which indicates students' progress on specific sub-steps of the problem. Feedback on subgoals offers special advantages because it demonstrates how a student can *break down a problem into a set of smaller sub-tasks*; which is a key to simplifying the learning process [37, 38], and can promote students' retention in procedural domains [35]. To generate such feedback, learning environments need to be able to do *subgoal detection*, which is the process of detecting when a student completes a key objective or sub-part of a programming task (e.g. receiving and validating user input). However, subgoal detection *during* problem-solving is known to be extremely challenging because it requires assessing students' *intended problem solving approach* rather than their *program output*. In other words, it is difficult to automatically evaluate whether a student completed a subgoal *in the middle* of problem-solving due to the many possible strategies that students can approach to solve a problem, even when using test cases or autograders.

Historically, to provide feedback on subgoals, learning environments have used expert-authored models, where human experts encode a set of rules to predict solution strategies that students might perform to complete a specific subgoal. While expert models can generate accurate feedback with interpretable explanations, they also require extensive human participation particularly for open-ended programming tasks, where it becomes unmanageable to capture every possible correct solution [59]. More recently, data-driven (*DD*) models, where the model learns rules from historical data, have become more prominent models. This is because *DD* models reduce the expert-authoring burden, and have the potential to be easily scaled to more problems and contexts. Moreover, *DD* models learn from multiple students' solutions, which makes it capture code situations that human experts cannot easily perceive, particularly in open-ended programming tasks [49, 45]. However, *DD* models are dependent on the quality of the data, and may have lower accuracy than expert models [59, 48]; and, therefore, may sometimes provide misleading feedback in practice [48]. Both of these models have strengths and weaknesses, and in this paper we propose an approach that takes advantage of both.

Samiha Marwan, Yang Shi, Ian Menezes, Min Chi, Tiffany Barnes and Thomas Price "Just a Few Expert Constraints Can Help: Humanizing Data-Driven Subgoal Detection for Novice Programming". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 68-80. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

We present a hybrid approach that leverages both a data-driven model and expert insights to detect subgoal completion during problem-solving block-based programs. Our hybrid model is based on three main steps. First, we used an unsupervised data-driven model to generate subgoal detectors for a programming task, and represent them as a set of human-*interpretable* and human-*editable* rules. Second, this representation allowed us to evaluate the accuracy of the subgoal detectors; particularly when they have inaccurate detections. Third, we used human expert insights to refine and fix rules that led to inaccurate subgoal detections. These three steps resulted in a *hybrid* data-driven approach that generates subgoal detectors with high accuracy, that target expert-authoring effort *only* where improvements are needed, and that can also be easily scaled to various problems and contexts.

We evaluated our hybrid data-driven model to a block-based programming problem from an introductory CS classroom against the same, fully data-driven (*DD*) model, *but* without experts' intervention. We evaluated the accuracy of both models by comparing their subgoal detections on a given programming task, to that of human experts, and we hypothesize that our hybrid model will surpass the accuracy achieved by the *DD* model. We found that the expert evaluations of subgoal detections achieved significantly higher agreement with our hybrid model than that achieved with the *DD* model. We also found that our hybrid model outperforms the state-of-the-art supervised models: code2vec [53], Support Vector Machine (SVM) [14], and XGBoost [9]. In addition, we present case studies of how the hybrid model led to differing subgoal detections in student programs compared to the *DD* model. We also discuss how we can close the loop by applying our hybrid model in block-based programming classrooms to provide students with immediate feedback on subgoals.

In summary, in this paper we investigate this research question: **RQ:** *How well does a hybrid data-driven model combined with expert insights perform compared to: 1) a data-driven model without expert augmentation and 2) baseline supervised learning approaches that leverage expert subgoal labels?* Our work provides the following contributions: (1) we present a hybrid subgoal detection approach which combines an unsupervised data-driven model with domain expertise to achieve the benefits of both data-driven models, and expert models, and (2) we demonstrate how our hybrid approach advances the state of the art in subgoal detection in open-ended programming tasks over supervised and unsupervised baselines, with an accuracy range of 0.80 - 0.92.

2. LITERATURE REVIEW

In this work, we investigate the challenge of automatically detecting subgoals effectively. We propose a method that involves a hybrid approach where human experts modify data-driven models to build effective subgoal detectors. In the following, we first review prior work on the immediate feedback with a focus on subgoal feedback. Then, we review prior work that involves merging machine and human expert intelligence to improve performance of machine learned models. Finally, we review both state-of-the-art supervised learning models and an unsupervised data-driven model that we used for subgoal detection in a programming task.

2.1 Feedback and Subgoal Detection

Formative feedback is defined as a type of task-level feedback that provides *specific*, *timely* information to a student in response to a particular problem, based on the student's current ability [54]. In a review of effective formative feedback in education, Shute shows that immediate formative feedback is effective because it can improve students' learning [11, 38] and retention [54, 44], particularly in procedural skills such as programming [54]. Most intelligent tutoring systems provide immediate feedback through identifying errors (e.g. error detectors [5, 55], anomaly detectors [31], or misconception feedback [25, 24]); however far less work has been devoted to providing feedback on students' *subgoals*. Automated assessment systems (i.e. autograders) can provide feedback on correct subgoals by showing the *passing* test cases using expert-authored models [7, 28, 26, 38]. For example, most autograders use instructor test cases to check for correct program output; however they require students to submit an almost-complete program to obtain feedback [7, 29]. As a result, this feedback is often not available in the early stage of programming when timely feedback on subgoals is mostly needed. To provide timely subgoal feedback, prior research has taken two exclusive approaches: *expert-authored* approach and *data-driven* approach.

Expert-authored Approaches: Prior work has explored student completion to subgoals by diagnosing students' solutions against expert models (e.g. constraint-based models [42]), even when a student has incomplete submissions, to provide feedback on whether they are on track [27], or whether they completed key objectives of short programming tasks [38]. However, these systems often require extensive expert effort to create rules. To address this authoring burden, example-tracing tutoring systems infer tutoring rules based on examples of potential student behaviors. This still requires an author with some domain expertise, but it allows rules to be constructed by non-programmers who have domain expertise [2]. An expert can create different example solutions to capture different solution strategies; and augment them with hints or feedback. Example-tracing tutors have been developed in multiple non-programming domains like genetics [12], mathematics [1], and applied machine learning, and they have been shown to improve the problem-solving process and student learning [2]. Despite the accuracy of expert models in providing feedback on test cases or correct features, which can be equivalent to subgoals, it is unclear how feasible they are in domains with vast solution spaces and open-ended problems, such as in programming tasks [59, 39, 25].

Data-driven Approaches: Data-driven approaches refers to systematically collecting and analyzing various types of educational data, to guide a range of decisions to help improve the success of students [15, 50]. Data-driven models largely avoid the need for expert authoring altogether by using prior students' correct solutions, instead of expert rules or instructor solutions, to learn patterns of correct solutions. This enables automated assessment feedback on student code [21]. Many data-driven models work by executing a comparison function that calculates the distance between students' code and all the possible correct solutions, and then compares the path of the most close solution with that of the student [47, 49, 59, 39]. While most data-driven methods have

been used to generate fine-grained feedback – such as hints in the hint factory [58], little work has used these methods to generate subgoal feedback. For example, Diana et al. developed a model that searches for meaningful code chunks in student code to generate data-driven rubric criteria to automatically assess students' code [19]. Diana et al show that a data-driven model can have agreement with that of the experts [19]. In the iList tutor, Fossati et al. used a data-driven model to provide feedback on correct steps, where they assess student code edits as *good*, if it improves the student's probability to reach the correct solution, or *uncertain* if a student spent more time than that taken by prior students at this point [21]. Fossati et al. found that this feedback was well-perceived by students, and improved their learning [21]. However, data-driven models are dependent on the similarity of the current student's approach to prior student submissions, making it difficult to control the quality of their feedback [39, 51, 59, 46]. In a case study paper, Shabrina et al. discuss the practical implications of data-driven feedback on subgoals, showing that the quality of feedback is important, even positive feedback, since inaccurate feedback can cause students to spend more time on a task even after they were done [51]. Because of such challenges and perhaps other reasons such as the inability for individual instructors to augment autograder feedback, few tools have been built to provide immediate feedback to students on whether they have achieved subgoals during their programming tasks [34].

2.2 Integrating Expert Knowledge into Models

In recent years, combining machine and human intelligence has been extensively explored in a wide range of domains including artificial intelligence and software engineering. For example, Chou et al developed a virtual teaching assistant (VTA) that uses teachers' answers as human intelligence, and machine intelligence to use teachers' answers to locate student errors, and generate hints [10]. Chou et al found that these mechanisms reduce teacher tutoring load and reduce the complexity of developing machine intelligence [10]. In the software engineering domain, there is an emerging approach called *collective intelligence* that merges the wisdom of multiple developers with program synthesis algorithms, which has been shown to significantly improve the efficiency and accuracy of program synthesis [61].

Human-in-the-loop methods are another effective trend that use human intelligence to improve the efficiency of Machine Learning models, *while* the model is learning [23, 56, 30, 63]. For example, Goecks introduces a theoretical foundation that incorporates human input modalities, like demonstration or evaluation, to leverage the strengths and mitigate the weaknesses of reinforcement learning (RL) algorithms [23]. Their results show that using human-in-the-loop methods accelerates the learning rate of RL models, with a more efficient sample, in real time [23]. Our work also uses human intelligence to improve the accuracy of a machine learning model; however, it does so *after* the model is trained.

In the educational domain, expert knowledge is widely applied to augment data-driven and machine-learned models for problem solving and feedback. For example, in a logic tutor that provides data-driven hints using students' solutions, Stamper et al. used an initial small amount of sample data generated by human experts to enhance the automatic delivery of hints [57]. Moreover, example-tracing tutors allow experts to specify moderately-branching solutions for open-ended problems, allowing some intelligent tutors *originally* implemented using complex expert systems to be almost completely replicated to support practical learning needs [2].

2.3 Supervised Learning Models of Code Analysis

Supervised learning algorithms leverage labels created by human experts, to guide the model search process. With available labels, automated learning algorithms can be applied to the subgoal detection tasks for programming data. As shown in [6], one baseline is to extract term frequency-inverse document frequency (TF-IDF) features and uses traditional machine learning algorithms such as support vector machines (SVM) [14] and XGBoost [9]. However, as word- or token-based features such as TF-IDF lose important structural information from programming data [3], recent work uses structural representations from code and a more complex model structure to learn more complex features. For example, Shi et al. applied a code2vec [4] model to detect the completion of rubrics on student programming data [53]. In this work, we compared our hybrid data-driven model to these existing supervised learning baseline models to check our improvement on the subgoal detection task.

2.4 Data-Driven Subgoal Detection Model

Among the various data-driven models for detecting subgoals, or rubric items [39, 18, 19], we built our proposed hybrid model on top of an unsupervised data-driven subgoal detection (*DD*) algorithm, presented in [64]. We applied this algorithm on a programming task called *Squiral* (described in Section 3) by running the following steps. First, the algorithm extracts prior student solutions in *Squiral* as abstract syntax trees (ASTs) [49, 47]. Then, it applies hierarchical code clustering for feature extraction and selection by: (1) extracting common **code shapes**, which are common subtrees, in ASTs of correct students' solutions (Figure 2 shows examples of code shapes); (2) filtering redundant code shapes; (3) identifying decision shapes, which consist of a disjunction of code shapes (i.e. $C_1 \wedge \ldots \wedge C_n$) that are usually mutually-exclusive (e.g. a program uses a loop, or a repeated set of commands, but not both), and (4) hierarchically clustering frequently co-occurring code shapes into *subgoals*. In [64], the authors found a meaningful Cohen's Kappa (0.45) in agreement of the algorithm and expert subgoal detection on student data, suggesting that *DD* subgoals could be used to generate feedback. However, since the *DD* subgoals are typically represented in a regular-expression-like form, labels are needed to make them meaningful and usable for students in programming environments.

3. METHOD

This work presents and evaluates a hybrid data-driven model for generating and detecting subgoals in a block-based programming exercise (explained in Section 3.1). To evaluate our hybrid data-driven approach, we applied our model on student data collected from an Introduction to Computing course, and we asked human experts to evaluate the accuracy of its subgoal detections (explained in Section 3.2.2).

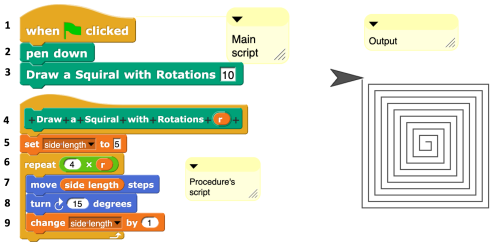


Figure 1: One possible solution for Squiral with line numbers on the left, and the script's output on the right.

We use this dataset to provide examples of how our approach works, but we also discuss how it can be generalized. We compared our hybrid model against its underlying data-driven (DD) model described above in Section 2.4, as well as state-of-the-art supervised learning models (explained below in Section 4.1).

3.1 Dataset

Our data is collected from a CS0 course for non-majors in a public university in the United States that uses Snap*!*, a block-based programming environment [22]. This programming environment logs all students actions while programming (e.g. adding, deleting or running blocks of code) with the *time* taken for each step. These student logs (i.e. actions) can also be replayed as a *trace* of code snapshots of all students' edits – allowing us to detect the time and the code snapshot where a subgoal is completed *during* student *problem-solving process*.

In this work, we collected students' logs from one homework exercise named *Squiral*, derived from the BJC curriculum [22]. *Squiral* requires a visual output, where students are asked to write a procedure that takes one input `'r'` to draw a square-like spiral with `r` rotations. Figure 1 shows a possible correct solution of *Squiral* that requires procedures, variables, and loops. We collected a training dataset from 3 semesters: Spring 2016 (S16), Fall 2016 (F16), and Spring 2017 (S17), which includes data of 174 students, that has a total log data of 29,574 student actions

3.2 Hybrid Data-Driven Subgoal Detection

Our hybrid data-driven model is based on two main things. First, the *DD* model is used to generate data-driven subgoals. Second, expert-authored constraints are added to improve the quality of these subgoals and the accuracy of their detection. We implemented our hybrid approach by conducting the following 3 high-level steps:

1. We used the *DD* model to generate an initial set of subgoals, consisting of clusters of code shapes. We then presented these clusters to experts in an interpretable form, who *combined* them to create a concrete set of meaningful subgoal labels.

2. We integrated *DD* subgoal detection model into the students' programming environment, allowing students to see when the *DD* algorithm detected completion of each subgoal. We then collected student programming log data along with *DD* detections, and asked experts to evaluate the accuracy of the *DD* detections.

3. We used human expert insights to fix code shapes that led to false subgoal detections; and then combined them again to create a modified set of hybrid subgoals and evaluated its new accuracy.

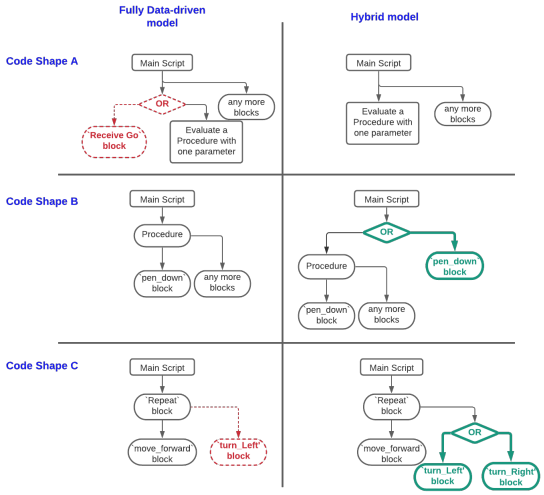


Figure 2: Three code shapes A, B, C, in both the data-driven and hybrid models. Each code shape represents a false detection and its fix by human experts. Red dashed nodes are removed and green bold nodes are added.

3.2.1 *Step 1: Interpreting and Editing Data-Driven Subgoals*

The goal of this step is to generate data-driven subgoals using the *DD* model and present them in an interpretable and editable form. We applied the *DD* model (described in Section 2.4) on S16, F16, and S17 student datasets to generate a number of clustered code shapes. Column 1 in Table 1 shows the description of 7 subgoals corresponding to code-shape clusters generated from *correct* solutions (n = 52). We evaluated each cluster by displaying its code shapes separately and *interpreted* their code behavior. For example, code shape A in Figure 2, on the left, represents a decision shape that requires student to use the `'ReceiveGo'` block (i.e. the hat block in Figure 1, line 1, which is used to start a script) in their main script, **or** to evaluate a procedure with one parameter, which is done by creating and snapping a procedure in the main script (as shown in Figure 1, line 3). We treated each cluster as a subgoal, and for a subgoal to be detected, the *DD* model requires all of its code shapes, and exactly one component of its *decision* shapes, to be present in student code.

While the data-driven clusters can represent appropriate subgoals, we combined some of them to create a shorter list of higher-level subgoals similar to the programming task rubric. Column 2 in Table 1 show the combined subgoals. It is worth noting that we also took the insights of two instructors of the CS0 course on how meaningful these subgoals are for students to understand. Additionally, because

one of our goals is to use these data-driven subgoals in learning environments, we asked human experts to label them to make them easily understandable by students, and instructors. For example, the human label of subgoal 1 is: "Create a procedure with one parameter, and use it in your code", as shown in Table 1. We then developed a data-driven subgoal detector that takes student code as an input, and outputs the status of each subgoal. For example, if the input is code C and the output is `{1, 0, 0, 1}`, this means the subgoal detector detects the completion of subgoals 1 and 4, and the absence of subgoals 2 and 3 in C.

3.2.2 Step 2: Investigating Data-Driven Subgoals

The goal of this step is to investigate the correctness of the *DD* subgoal detections. In Spring 2020 (S20), we integrated the *DD* subgoal detector into the Snap*!* programming environment for the *Squiral* exercise to provide students with subgoal feedback [39]. In Snap*!*, students could see the subgoal labels (shown in Table 1), colored gray to start, green when the subgoal was detected, or red when it became broken. We collected 4,480 edit logs from 27 student submissions with an average of 166 edits per student in S20. For each student edit, we recorded the *DD* subgoal detection state (e.g. `{1, 0, 0, 0}` for subgoal 1 being complete). We asked human experts to manually replay each student's trace data and evaluate whether the subgoal labels, as shown to students, were achieved at the specific timestamps when the *DD* algorithm detected them. Importantly, we asked experts to report when the expert-authored *labels* for each subgoal (which students saw) were achieved. Since these labels do not *precisely* match the code shape combinations that the *DD* subgoal detector used, it was very possible for the *DD* model to be "wrong." In other words, we asked experts to determine when each student achieved "Create a *Squiral* procedure with one parameter and use it in your code," and compared that to when the *DD* detector marked this subgoal label as complete.

We classified each evaluated instance as either `Early`, `onTime`, or `Late`. An instance is classified as `Early` if the *DD* detection is *before* the human expert detection timestamp, `OnTime` if they coincide, and otherwise `Late`. For example, if for student S_j, the human expert detected the completion of subgoal `i` (SG_i) at time `t` $= 5$; while the algorithm detected it at `t` $= 2$, then we label SG_i for S_j as `Early` detection. Then we sorted students in descending order based on the percentage of false detections in their log data, and we took the first 66% of this data ($\sim$ 18 students as a data sample) to investigate the reasons for false detections. We did not use the full set of false detections, since our primary goal was to fix the most common mismatches, without overfitting to the dataset.

We then focus on false detections that occured due to new, correct solutions, in the S20 dataset, that had no matching code shapes in the training dataset (S16, F16, and S17 datasets). We do not investigate expected false detections that occured due to known limitations in the *DD* algorithm (e.g. the *DD* algorithm does not differentiate between variable names).

We found three reasons for false detections for subgoals 1, 2, and 4. Inspired by the design of the constraint-based SqlTutor by Mitrovic et al. [41], we introduce 3 fixes (or constraints) to resolve them.

Subgoal 1 false detection. As shown in Table 1, subgoal 1 label requires a student to create a procedure with one parameter, and use it (or evaluate it) in the main script. However, subgoal 1 code shapes consist of the creation and evaluation of a procedure, ***or*** the use of a `'ReceiveGo'` block (the hat block used to start a script). This means that whether a student created and evaluated a procedure, *or* added a `'ReceiveGo'` block in the main script, the *DD* model will detect the completion of that subgoal, but experts did not interpret the 'ReceiveGo' block as meeting this subgoal, yielding a false detection. To fix this false detection, we simply removed the 'ReceiveGo' block as an option for this subgoal. Code shape A in Figure 2 shows the code shapes of subgoal 1 of the *DD* model (on the left), and how it is fixed in the hybrid model (on the right).

Subgoal 2 false detection. As shown in line 6 in Figure 1, subgoal 2 requires a student to use a 'repeat block that iterates 4 times the number of rotations to draw a *Squiral* with the correct number of sides. While code shapes of this subgoal satisfy this definition, they also include a code shape of adding a 'pen down' block, which is necessary to draw, but *only* inside a procedure. Therefore, if a 'pen down' block is used outside of a procedure, subgoal 2 will not be detected. To fix this false detection, we added a disjunction code shape to detect the presence of `'pen down'` inside *or* outside a procedure, as shown in code shape B in Figure 2.

Subgoal 4 false detection. As shown in lines 6-9 in Figure 1, subgoal 4 requires the use of 'move', 'turn', and 'change - by -' blocks (which increments a value of a variable), inside a 'repeat block'. We found that code shapes of subgoal 4 only include the `'turnLeft'` block; however, if the student solution contains a `'turnRight'` block (which does the same 'turn' functionality but from a different direction), the subgoal will not be detected. To fix this false detection, we modified all the code shapes that require the use of `'turnLeft'` block to accept either `'turnRight'` or `'turnLeft'` blocks, as shown in code shape C in Figure 2.

These three false detections show that prior solutions in S16, F16, and S17 datasets often used a `'ReceiveGo'` and `'turnLeft'` blocks, and used `'pen down'` inside a procedure; but this was not always the case in the S20 data. This shows that investigating the accuracy of a model, either data-driven or expert, is necessary since it is impossible to predict how students will behave in practice or how the data will change from one class to another.

3.2.3 Step 3: Improving the Data-Driven Subgoals with Human Insights

The goal of this step is to apply the human expert constraints (explained in Step 2) to mitigate the false detections of the *DD* algorithm. To do so, we developed a tool that iterates over each code shape of the data-driven subgoals, and allows humans to edit code shapes (i.e. add, delete or modify) to apply the three constraints (i.e. fixes) explained in Step 2. Because this tool modifies the code shapes, we then use the original *DD* algorithm to re-cluster all code shapes to ensure that the most similar code shapes remain

Table 1: Data-driven subgoals (composed of code shape clusters) with their corresponding human labels that were used when presented to students.

Data-Driven Code Shape Clusters	Combined Clusters	Subgoal Human Label
C1: Evaluate a procedure with one parameter on the script area OR Add a 'ReceiveGo' block.	C1 + C2 = Subgoal 1	Create a Squiral procedure with one parameter and use it in your code.
C2: Create a procedure with one parameter.		
C3: Have a 'multiply' block with a variable in a 'repeat' block OR two nested 'repeat' blocks.	C3 + C4 = Subgoal 2	The Squiral procedure rotates the correct number of times.
C4: Have a 'pen down' block inside a procedure		
C5: Add a variable in a 'move' block inside a 'repeat' block.	C5 = Subgoal 3	The length of each side of the Squiral is based on a variable.
C6: Have a 'move' and 'turnLeft' block inside a 'repeat' block.	C6 + C7 = Subgoal 4	The length of the Squiral increases with each side.
C7: 'Change' a variable value inside a 'repeat' block.		

clustered together. Figure 2 shows an example of three code shapes before and after they have been edited by human experts, that fix false detections that existed for subgoals 1, 2, and 4, respectively.

In summary, our hybrid model **humanizes** data-driven subgoal detection models through a series of important steps. First, we apply a data-driven model to correct, historical student solutions to generate a set of human *interpretable* code shape clusters. Second, a human expert labels the subgoals these clusters represent, in a way that is meant to align with the original programming assignment. Third, we collect data from students solving the same task using a programming environment augmented with subgoal feedback (i.e human labels with colors) based on the *DD* subgoal detector. Fourth, we had experts examine code traces with the *DD* subgoal feedback to determine when the displayed subgoal labels were actually achieved. Fifth, human experts modified the code shapes that led to discrepancies between the data-driven and expert detections for the displayed subgoals. This series of steps leverages the natural cycle of a frequently-offered CS0 class to bootstrap the creation of *DD* subgoal detectors in the programming environment.

4. EVALUATION

In this experiment, we applied both the hybrid and the *DD* models to detect subgoals in students' S20 code submissions. We also asked two human experts to evaluate the presence or absence of each subgoal in a subset of students' code snapshots (sequential states of student code, corresponding to their code edits, e.g., the addition or deletion of code blocks) using the subgoal labels (shown in Table 1) as rubric items (with 1 for the subgoal's presence and 0 for its absence), resulting in an *expert (or gold standard) subgoal state*.

Because S20 data consists of 4480 code snapshots, it is not feasible to evaluate the models on every timestamp for two reasons. First, students mostly need feedback on a given subgoal when they are making edits towards finishing *that subgoal*, not after every single edit they make. Second, students break and recomplete subgoals frequently, even when they are not working on a particular subgoal, and therefore it is not meaningful to have an expert label at every single datapoint. As a result, we evaluate the models at the most meaningful times when a student is close to finishing a subgoal, including: (1) the first time a student completed a subgoal, according to a human expert, (2) up to five code edits before that subgoal is completed, and (3) any time when either model (i.e. hybrid, or *DD*) suggests a *change* in a subgoal's status. While these changes may or may not be true, we wanted to have experts evaluate the correctness of how the algorithms may have detected subgoal changes at these points.

For each subgoal, two human experts evaluated 150, 163, 178, and 196 student snapshots for subgoals 1, 2, 3, and 4, respectively, making a total of 687 labeled snapshot datapoints. The experts used the subgoals as their rubric; and they started the labelling process by evaluating *the first time* a subgoal is detected. To do so, they divided the data (27 students * 4 subgoals = 108 datapoint) into a set of 6 rounds, where the first round consists of 2 students and the remaining 5 rounds consists of 5 students. The two experts collaboratively evaluated the first round together to make sure they have a clear understanding of the rubric subgoals. Then for the next two rounds, they evaluated the logs independently and after each, they met to discuss and resolve conflicts. For these 3 rounds, the two experts had a moderate to substantial agreement with a Cohen's kappa ranging from 0.5 to 0.67. The reason why the kappa values are low is that we considered any disagreement even if it was a difference of *one* timestamp, but it does highlight how subgoal detection can be subjective, which is a challenge for measuring the accuracy of subgoal detection. As a result, for the remaining data, the two experts continued to evaluate it independently and then met to discuss and resolve conflicts to produce relatively objective *gold standard* expert labels. We used these labelled logs as ground truth to compare the accuracy and F1-scores of both the *DD* and the hybrid models. We also calculated the agreement between the expert detections and those generated by the hybrid and *DD* models.

4.1 Supervised Learning Models

We also compared our hybrid humanized model with supervised machine learning models as another form of baseline. The supervised models were trained and tested on the S20 dataset, using the same 687 expert-labeled snapshots described above. We trained separate models to detect each of

the subgoal labels (using training/testing splits discussed below). This allows us to directly compare these supervised methods, which require *labeled* training data, to the *DD* model, which does not, and to our hybrid approach, where the expert uses *some* of the labeled data to improve the model. While we discuss some limitations to this comparison in Section 7, these baselines help contextualize the performance of our subgoal detectors.

The baseline supervised machine learning models we have chosen are two shallow models (SVM [14] and XGBoost [9]) and one deep learning model (code2vec [4]). We used the same edits for predictions as the *DD* and hybrid models, and extracted the term frequency–inverse document frequency (TF-IDF) features from the models, thus a vector representation of an edit is generated, and used for training the two models. We performed grid search cross validation for both models. For SVM, we used a parameter space of linear kernel and Radial Basis Function (RBF) [60] kernel, and searched the regularization parameters from 1 to 10. For XGBoost, we searched the subsampling space of 0.1 to 1, with the number of estimators from 5 to 100, stepping by 5. Ten-fold cross validation is performed to search the parameter spaces. The training, validation, and test sets are split by students to make sure that no students used for testing will have an edit in training, since edits from one student would be very similar, and using samples similar to the testing set in training would lead to an unfair comparison.

We selected one state-of-the-art deep learning model, code2vec [4], for comparison as well, as the model has recently been applied in educational code classification tasks [53]. Instead of using a vector of term frequency to represent edits, code2vec uses the structural representations from ASTs to represent the code, and the representation is learned from training a neural network[1]. We used early stopping to avoid overfitting. To ensure the robustness of our results, we ran 20 times with resampling for all supervised baseline models, and reported average metrics (e.g. F1-score, accuracy).

5. RESULTS

RQ1a: *How well does a hybrid model perform compared to a data-driven model without expert augmentation?*

The prediction results for each subgoal from the *DD* model and the hybrid model are shown in Table 2. Our hybrid model achieves higher accuracy and F1-scores on all subgoals than the *DD* model. In particular, it reaches > 0.8 accuracy for all subgoals, and it reaches > 0.9 accuracy for 2 out of the 3 subgoals that were modified (i.e. subgoal 1 and 4) with expert constraints. It is worth noting that the hybrid model achieves higher accuracy in subgoal 3, which was not modified with expert constraints. This is possible because after we modified code shapes for the *other* subgoals, we reclustered the code shapes (as described in Section 3.2.3), and the new code shapes for subgoal 3 were changed. This is likely because, after reclustering, some code shapes moved to subgoal 3, resulting in higher recall.

We also measured the agreement between human experts, *DD*, and hybrid model subgoal detections. For the four subgoals, we find low to moderate agreement over all student logs between *DD* and human expert detections, with Cohen's kappa values ranging between 0.25-0.581. However, we find substantial or better agreement between the hybrid model and human expert detections, with Cohen's kappa values ranging between 0.6-0.84. It is worth noting that this agreement is higher than that achieved between the two human experts (described in Section 4). These results showed that the addition of just three human constraints to a data-driven model succeeded in improving its accuracy, making the hybrid model agree *more* with the gold standard (that the experts co-constructed), than the experts' original agreement with one another.

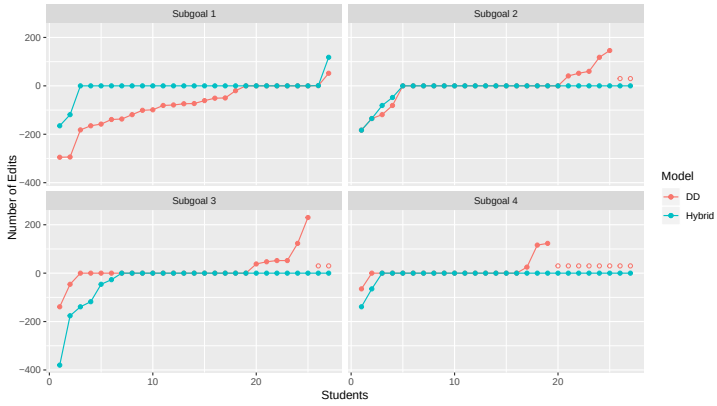


Figure 3: The number of code edits (y-axis) that occurred between gold standard expert subgoal detections, and detections by the DD and Hybrid models, for first-time subgoal detections for each student trace (x-axis).

We also determined the number of false detections (i.e. `Early` and `Late` detections, as described in Section 3.2.2) for both the hybrid and *DD* models. We found the *DD* model detected 40.66%, 10.66%, 5.62% and 5.10% `Early` detections, and 2%, 13.5%, 12.36%, and 15.31% `Late` detections for subgoal 1, 2, 3, and 4, respectively. However, our hybrid model detected 1.33%, 13.5%, 10.67% and 4.6% `Early` detections, and 8%, 5.52%, 1.7%, and 2.81% `Late` detections for subgoal 1, 2, 3, and 4, respectively. To visualize these false detections, Figure 3 visualized these false detections by presenting the distance (i.e. how many edits) between the `Early` and `Late` detections by both the *DD* and hybrid models, and the gold standard human expert detections of the *first time* a subgoal is completed. The x-axis presents the students ($n = 27$), and the y-axis presents the number of edits a student makes until they complete a subgoal. We used a negative number to indicate how much *earlier* the models were than the gold standard detection, 0 to show when models *agree* with the gold standard, and a positive number to show how much *later* the models were. We also used *empty* circles to indicate instances where a subgoal is *never* detected by the models but it was detected by human experts. While Figure 3 shows *only* how early/late the model is in detecting when a subgoal is *first* completed, this is likely the most important detection. Our results suggest a high agreement between the hybrid model's detections with the gold standard, and a strong improvement over the *DD* model.

[1]We applied code2vec using the process described in [53].

Table 2: Precision, Recall, F1-score and Accuracy observed with Supervised, Data-Driven (DD) and our Hybrid Models.

	Precision					Recall					F1-score					Accuracy				
	SVM	XG-Boost	Code2vec	DD	Hybrid	SVM	XG-Boost	Code2vec	DD	Hybrid	SVM	XG-Boost	Code2vec	DD	Hybrid	SVM	XG-Boost	Code2vec	DD	Hybrid
Subgoal 1 (n = 150)	0.71	0.68	0.89	0.44	**0.95**	0.79	0.75	0.91	**0.94**	0.76	0.71	0.66	**0.89**	0.60	0.85	0.82	0.76	0.90	0.57	**0.91**
Subgoal 2 (n = 163)	0.58	0.61	0.57	**0.70**	0.69	0.61	0.77	0.79	0.63	**0.85**	0.55	0.66	0.64	0.66	**0.76**	0.74	0.75	0.75	0.77	**0.81**
Subgoal 3 (n = 178)	0.60	0.62	0.69	**0.80**	0.75	0.54	0.61	0.76	0.64	**0.95**	0.52	0.59	0.70	0.71	**0.84**	0.70	0.72	0.79	0.82	**0.88**
Subgoal 4 (n = 196)	0.62	0.64	0.64	0.79	**0.87**	0.64	0.75	0.84	0.55	**0.93**	0.59	0.66	0.69	0.65	**0.90**	0.76	0.74	0.75	0.80	**0.93**

RQ1b: *How does a hybrid model perform compared to supervised learning approaches that leverage expert subgoal labels?*

We show our comparison of supervised learning models and the hybrid model in Table 2. On all subgoals, except one, the hybrid model has higher accuracy and F1-score than code2vec, SVM, and XGBoost models, outperforming them by 0.10, 0.06 and 0.15 percent of F1-score, respectively. In subgoal 1, we found that code2vec achieved a higher F1-score than all the other models, and a relatively similar accuracy to the hybrid model (0.903, 0.906). One possible explanation for this is that subgoal 1 is the simplest subgoal, requiring only that the student has defined and used a procedure, regardless of its content, and this simple code pattern may have been easier for the supervised approaches to learn. These results show that a hybrid model iteratively constructed through cycles of student data collection, machine learning, along with human labeling and correction can be used to create accurate automatic subgoal detections on a novice programming task. Furthermore, these supervised learning models, that were mostly outperformed by our hybrid model, were learned using labels from snapshots that were *strategically chosen* to reflect important decision points for the model, suggesting that the supervised models' performance may suffer if a random selection of snapshots were used to create an expert-labeled training set instead.

5.1 Case Studies

In this section, we present case studies to highlight ways the hybrid model improved upon the original *DD* model, as well as the hybrid model's limitations. These case studies come from the 33% of students who were *not investigated* when the expert identified false detections in S20 from the original *DD* model, as discussed in our methods (Section 3.2.2). These students also used the original *DD* system, but their data did *not* inform our hybrid model. These case studies, therefore, help us understand the ways our hybrid model might help new students, as well as limitations of the model. Though our prior work suggests the *DD* subgoal detections overall were often useful to students [39], our post-hoc analysis here shows that the *false detections* may have negatively impacted student programming behavior, suggesting the need for our hybrid model's improvements.

5.1.1 Case Study 1 (Em): Inaccurate Data-Driven Subgoal Feedback

We present here a case study of the student Em[2] when they received an inaccurate subgoal detection based on the *DD* model, and how the hybrid model could have mitigated this false detection.

Em started solving *Squiral* by snapping the 'when green flag clicked block' (i.e. 'ReceiveGo' block) on the main script as shown in Figure 4A, and the system *falsely* detected subgoal 1. Em then proceeded to work on subgoal 2, *without* creating the required procedure, and created a loop using the 'repeat' block nested with 'move' and 'turnLeft' blocks (shown in Figure 4B). This time, the system was correct in not detecting subgoal 2 because the loop was not in a procedure, and does not iterate on the 'rotations' parameter. Afterwards, Em correctly created a procedure with one parameter as shown in Figure 4C; however, the system shows no change, since it already falsely detected subgoal 1 earlier, and therefore, no change in the feedback is given to the student. Em then destroyed the procedure, without ever making it again. Em kept working for the rest of the time on creating a number of redundant loops, similar to the one in Figure 4C, with constant values to manually draw the *Squiral* shape (rather than using a variable to vary its length).

Em spent a total of 55 minutes to draw *Squiral* in an iterative manner. While the *DD* system accurately detected subgoals 2-4 as incomplete, this case study highlights potential harm that may have arisen from the false detection of subgoal 1. When subgoal 1 was detected early, Em skipped over creating a procedure. Later, when she did create the procedure correctly, she got no additional feedback (since the subgoal was already detected), and promptly deleted it. Preventing these unneeded deletions is a primary role of correct, positive subgoal feedback. However, had Em been using the hybrid model, subgoal 1 would not have been detected early because the expert edited the faulty code shape. We argue that this might have allowed Em to keep working on creating a procedure (as shown in snapshot C in Figure 4), which would have been detected as complete by the hybrid system only at this time. It is also possible that receiving inaccurate feedback at the very beginning may have led to Em's mistrust in the system, since prior work shows that incorrect feedback can reduce students' willingness to use it [48].

Note that, we do not believe this incorrect *DD* detection

[2]We provide anonymous names for students.

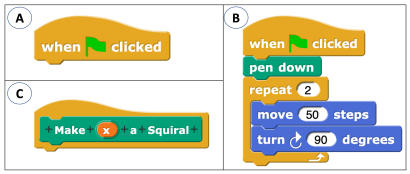


Figure 4: Code snapshots A, B, and C, implemented by student Em in Case Study 1.

means the full data-driven system should not be used since our prior work shows the system can be helpful to students [39]. However, we need to explore how to present subgoal feedback in a way that promotes students to question the feedback since any such program will inevitably fail to recognize some correct variants of a subgoal solution.

5.1.2 *Case Study 2 (Jo): Cases when Hybrid Subgoal Detections could be Incorrect*

We present here a case study where the hybrid model would *not* have provided accurate subgoal feedback. As evidenced by the model's high overall accuracy (Table 2), these instances were rare, but understanding them highlights the affordances and limitations of our approach. Specifically, we investigated subgoal 1, where the hybrid model had the lowest F1-score.

Student Jo correctly created a procedure with a parameter; however, since Jo had not yet used the procedure in their main script, subgoal 1 was not detected (accurate detection). Jo continued programming and completed subgoals 2 and 3, which were accurately detected by the system. Afterwards, Jo added a procedure call to the main script, and subgoal 1 was detected as complete (accurate detection). However, Jo then added a 'pen up' block underneath the procedure call, where unexpectedly, the hybrid model changed subgoal 1's status back to incomplete (incorrect detection).

This false detection in the hybrid model was due to an overly-specific code shape. Specifically, the code shape required the procedure call to be the *last block* in the main script (which was true for 94% of students, but not Jo), leading to the false detection. This case confirms the importance of iteratively investigating and refining data-driven subgoal detections to keep improving their accuracy, which is a common process in expert-authored models as well. While this false detection has a straightforward fix, similar to the ones presented in Section 3.2.2, it shows one limitation of the hybrid model: creating these fixes requires the expert to find and address the false detections in the first place, which is dependent on finding the bugs in the data inspected. This is also one reason why the hybrid model performance of subgoal 1 has a lower F1-score than the code2vec model (as shown in Table 2).

6. DISCUSSION

6.1 Automated Subgoal Detection

The key contribution of this paper is tackling the critical challenge of *automated subgoal detection* during programming tasks. Our results show that a hybrid data-driven model meaningfully addresses this goal, with high accuracy and F1 score when detecting subgoals at key moments during students' work. Our results show that this is a challenging task: even a state-of-the-art supervised learning approach with access to labeled data struggled to identify some subgoals (F1 score as low as 0.64). This agrees with prior work using expert-authored [13, 38] and supervised learning models [53], showing that immediate feedback on subgoals is a hard problem.

While automatically detecting subgoals is challenging, research suggests that the ability to provide automated, immediate feedback on subgoals can significantly improve students' motivation and learning. Providing subgoals for novices can improve student learning by breaking down the programming task into smaller subtasks, which is a challenging task for novices [36, 38]. In human tutoring dialogues for programming, tutors provide a combination of corrective and positive feedback, increasing students' motivation and confidence in programming [8, 33, 17]. Automatic subgoal detection could be used to provide similar corrective and positive feedback during programming. We know of only 3 systems that can afford such immediate feedback, that is not based on unit tests, during programming, that have been shown to promote learning, confidence and persistence for linked lists [21], database queries [42], and block-based programming [38]. It is perhaps uncommon to make such systems due to the difficulty in anticipating all student approaches, paired with the high potential for inaccuracies and student reactions to them. Our accuracy results suggest that our hybrid, humanized approach can be used to build similar automatic subgoal detection systems that could be deployed and more easily scaled across problems in real classrooms.

6.2 Affordances of Data-driven, Hybrid and Expert models

Our results suggest that the hybrid model has good potential for solving the problem of automated subgoal detection. Here we discuss the advantages and trade-offs of the hybrid approach, compared to data-driven and expert models.

6.2.1 *RQ1.a: Hybrid versus Data-Driven Models*

Our results show that a hybrid, iterative model that leverages data-driven subgoal extraction, human labeling, and expert refinement based on labeled student data, can *greatly* improve model performance compared to a purely data-driven (DD) model. The expert constraints improved F1-score of the data-driven model by 0.14-0.25 points, as shown in Table 2. Based on our analysis, the hybrid model, reduced the number of `Early` and `Late` subgoal detections and increased the `onTime` detections, when compared to the original *DD* model, as shown in Figure 3. This is a critical improvement, since prior work shows that the quality of feedback affects novices' programming behavior [48, 51], but also their self-perceptions, and trust in the learning environment [48].

The hybrid model creation does require additional labelling effort needed to evaluate the models; however, this effort seems well worth it, and is needed to evaluate the accuracy of any data-driven model before deployment. Compared to prior work, our iterative hybrid model shares similar benefits of "human-in-the-loop" methods in machine learning [56, 30] and also represents data-driven rules in an interpretable and

editable form that simplified the process of merging human insights into the model. In prior work, Diana et al. found, qualitatively, that their generated data-driven rubrics are considerably similar to human-generated rubrics [19]. Similarly, in our work, not only did instructors agree with the hybrid model subgoal detections, we also found these detections have substantial or better agreement with the human expert gold standard than the *DD* model. From these results, we conclude that **a hybrid model can be used to iteratively improve and humanize data-driven subgoal detection.**

6.2.2 RQ1b: Hybrid versus Supervised Learning Models that Leverage Expert Subgoal Labels

Our results show that **a hybrid model can surpass the performance of supervised learning models.** The steps we used to create our hybrid model were to: (1) apply a data-driven model, (2) add expert constraints, and (3) determine interesting datapoints consisting of times when subgoals might be achieved. The steps we used to create supervised learning models leveraged the interesting datapoints from step 3 of our hybrid model, hand-labeled them, and used them to build supervised learning models. We highlight this to point out that it is hard to determine what labeled data to use in a supervised learning model for subgoal detection, since there are hundreds of potential snapshots from each student. As a result, it is unclear whether the supervised model would have performed as well, compared to one learned from a labeled dataset selected at random or regular timestamps. Our results show that, even after using carefully selected labelled data to train the model, the SVM and XGBoost baselines did not achieve the level of accuracy of the hybrid model for all subgoals. Even code2vec, the state-of-the-art supervised learning model [53], has lower performance for all subgoals, except subgoal 1, than the hybrid model. Perhaps the code2vec performed worse due to the size of labelled dataset (687 datapoints), though recent results suggest the model is still effective with small datasets [52].

6.2.3 Hybrid versus Expert Models

Our hybrid approach offers distinct advantages and trade-offs compared to expert-authored models for subgoal detection. Traditional expert-authored models (e.g. constraint-based tutors [42]) have the advantage of high accuracy and high confidence, but the trade-offs of considerable domain expert time for creation and the potential failure to anticipate some student strategies and misconceptions. Our hybrid model has the advantage of incorporating actual student strategies and misconceptions, and primarily requires human effort to *label data* and identify errors, tasks which can potentially be done by non-experts and distributed across multiple people. A domain expert is only needed to edit the automatically extracted rules and is afforded the chance to do so with actual student data available. A significant tradeoff of the hybrid model is its reliance on data - so the quality of the dataset will directly impact the quality of the subgoal detectors. Furthermore, both models are likely to need refinement as students use them, and this process is already built into the hybrid model creation and refinement cycle.

7. LIMITATIONS & CONCLUSION

This work has 5 main limitations. First, while the *DD* model can capture small differences in solution approaches in *Squiral* (like having whether a 'turnLeft' or 'turnRight' block), we have not tested it in programming tasks with larger space of solution approaches. Therefore, it is not known how well the accuracy results will generalize to other types of programming tasks or languages. However, the iterative process of data collection, *DD* subgoal extraction, labeling, and collection of data from students using the subgoal labels and detectors, could be applied for other programming problems, of the same level as *Squiral*, and repeated until the models achieved high accuracy. Second, some of S20 data that was used for models' evaluation was also used to inform expert constraints in the hybrid model. However, this was only 66% of the data, and we discussed above how the added constraints are generalizable, which should have helped in any semester (see Section 5.1). Additionally, our case studies in Section 5.1 show examples of how the hybrid detector performed on unseen data, though there was insufficient data for a quantitative evaluation.

Third, we used only the labelled S20 data to train the supervised baseline models, but we also used 3 other semesters of *unlabeled* data to train the unsupervised *DD* and hybrid models. However, we argue that this ability to leverage a larger unlabeled dataset is an advantage of the unsupervised methods, rather than a limitation of our analysis. Fourth, some of the datapoints that were labeled for the evaluation of all the models were selected in part by using the hybrid and *DD* model detections, as discussed above, and this might have biased the results. However, all the models were evaluated on these same datapoints that were *strategically* chosen for their importance, and there are instances where some of the supervised models outperformed the original *DD* model. It is not clear how a different data selection strategy would have affected the results, and we argue that training and testing the supervised models on a dataset of the same size with randomly selected snapshots would likely decrease the performance of supervised models. Finally, we did not compare the hybrid model to a purely expert-authored model, and we did not measure time taken by experts to modify the data-driven rules. We argue that these comparisons require hiring experts to author rules and performing time analysis, which is beyond the scope of this paper.

In summary, this work proposes a new paradigm for 'humanizing' data-driven subgoal detection for novice programming. Specifically, we proposed to humanize data-driven subgoals in an iterative refinement process. We (1) extract data-driven subgoals from student work, (2) give them human labels, (3) collect more data from students programming with the labels and subgoal detectors, (4) present experts with the labels, and interpretable detectors, along with student behavior data so they can add expert constraints. This process ensures that humans are involved in every step of the creation of automatic subgoals, offering the advantages of reflecting real student behaviors, and limiting and focusing expert authoring time. Our results show that this hybrid humanized model outperforms fully data-driven models and state-of-the-art supervised learning models. This proposed paradigm can be used to create humanized automatic subgoal detection for tasks where it may be too expensive to create full expert models for, but that are important for student learning, motivation, and retention.

8. REFERENCES

[1] V. Aleven, B. M. McLaren, and J. Sewall. Scaling up programming by demonstration for intelligent tutoring systems development: An open-access web site for middle school mathematics learning. *IEEE transactions on learning technologies*, 2(2):64–78, 2009.

[2] V. Aleven, B. M. McLaren, J. Sewall, M. Van Velsen, O. Popescu, S. Demi, M. Ringenberg, and K. R. Koedinger. Example-tracing tutors: Intelligent tutor development for non-programmers. *International Journal of Artificial Intelligence in Education*, 26(1):224–269, 2016.

[3] U. Alon, M. Zilberstein, O. Levy, and E. Yahav. A general path-based representation for predicting program properties. *PLDI'18*, 2018.

[4] U. Alon, M. Zilberstein, O. Levy, and E. Yahav. code2vec: Learning distributed representations of code. *POPL'19*, 2019.

[5] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207, 1995.

[6] D. Azcona, P. Arora, I.-H. Hsiao, and A. Smeaton. user2code2vec: Embeddings for profiling students based on distributional representations of source code. In *LAK'19*, 2019.

[7] M. Ball. Lambda: An Autograder for *snap*. Technical report, Electrical Engineering and Computer Sciences University of California at Berkeley, 2018.

[8] K. E. Boyer, R. Phillips, M. D. Wallis, M. A. Vouk, and J. C. Lester. Learner characteristics and feedback in tutorial dialogue. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 53–61. Association for Computational Linguistics, 2008.

[9] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4), 2015.

[10] C.-Y. Chou, B.-H. Huang, and C.-J. Lin. Complementary machine intelligence and human intelligence in virtual teaching assistant for tutoring program tracing. *Computers & Education*, 57(4):2303–2312, 2011.

[11] A. Corbett and J. R. Anderson. Locus of Feedback Control in Computer-Based Tutoring: Impact on Learning Rate, Achievement and Attitudes. In *Proceedings of the SIGCHI Conference on Human Computer Interaction*, pages 245–252, 2001.

[12] A. Corbett, L. Kauffman, B. Maclaren, A. Wagner, and E. Jones. A cognitive tutor for genetics problem solving: Learning gains and student modeling. *Journal of Educational Computing Research*, 42(2):219–239, 2010.

[13] A. T. Corbett and J. R Anderson. Knowledge decomposition and subgoal reification in the act programming tutor. 1995.

[14] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] S. Custer, E. M. King, T. M. Atinc, L. Read, and T. Sethi. Toward data-driven education systems: Insights into using information to measure results and manage change. *Center for Universal Education at The Brookings Institution*, 2018.

[16] J. Denner and L. Werner. Computer programming in middle school: How pairs respond to challenges. *Journal of Educational Computing Research*, 37(2):131–150, 2007.

[17] B. Di Eugenio, D. Fossati, S. Ohlsson, and D. Cosejo. Towards explaining effective tutorial dialogues. In *Annual Meeting of the Cognitive Science Society*, pages 1430–1435, 2009.

[18] N. Diana, M. Eagle, J. Stamper, S. Grover, M. Bienkowski, and S. Basu. Data-driven generation of rubric parameters from an educational programming environment. In *International Conference on Artificial Intelligence in Education*, pages 490–493. Springer, 2017.

[19] N. Diana, M. Eagle, J. Stamper, S. Grover, M. Bienkowski, and S. Basu. Data-driven generation of rubric criteria from an educational programming environment. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*, pages 16–20, 2018.

[20] M. L. Epstein, A. D. Lazarus, T. B. Calvano, K. A. Matthews, R. A. Hendel, B. B. Epstein, and G. M. Brosvic. Immediate feedback assessment technique promotes learning and corrects inaccurate first responses. *The Psychological Record*, 52(2):187–201, 2002.

[21] D. Fossati, B. Di Eugenio, S. Ohlsson, C. Brown, and L. Chen. Data driven automatic feedback generation in the ilist intelligent tutoring system. *Technology, Instruction, Cognition and Learning*, 10(1):5–26, 2015.

[22] D. Garcia, B. Harvey, and T. Barnes. The beauty and joy of computing. *ACM Inroads*, 6(4):71–79, 2015.

[23] V. G. Goecks. Human-in-the-loop methods for data-driven and reinforcement learning systems. *arXiv preprint arXiv:2008.13221*, 2020.

[24] L. Gusukuma, A. C. Bart, D. Kafura, and J. Ernst. Misconception-driven feedback: Results from an experimental study. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*, pages 160–168, 2018.

[25] L. Gusukuma, D. Kafura, and A. C. Bart. Authoring feedback for novice programmers in a block-based language. In *2017 IEEE Blocks and Beyond Workshop (B&B)*, pages 37–40. IEEE, 2017.

[26] P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli calling international conference on computing education research*, pages 86–93, New York, NY, 2010. ACM.

[27] J. Jeuring, L. T. van Binsbergen, A. Gerdes, and B. Heeren. Model solutions and properties for diagnosing student programs in ask-elle. In *Proceedings of the Computer Science Education Research Conference*, pages 31–40, 2014.

[28] D. E. Johnson. Itch: Individual testing of computer homework for scratch assignments. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 223–227. ACM, 2016.

[29] D. E. Johnson. Itch: Individual testing of computer homework for scratch assignments. In *Proceedings of the 47th ACM Technical Symposium on Computing*

Science Education, pages 223–227, New York, NY, 2016. ACM.

[30] B. Kim. *Interactive and interpretable machine learning models for human machine collaboration*. PhD thesis, Massachusetts Institute of Technology, 2015.

[31] N. Körber, K. Geldreich, A. Stahlbauer, and G. Fraser. Finding anomalies in scratch assignments. *arXiv preprint arXiv:2102.07446*, 2021.

[32] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen. A study of the difficulties of novice programmers. *Acm Sigcse Bulletin*, 37(3):14–18, 2005.

[33] M. R. Lepper, M. Woolverton, D. L. Mumme, and J. Gurtner. Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. *Computers as cognitive tools*, 1993:75–105, 1993.

[34] A. Luxton-Reilly, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard, and C. Szabo. Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 55–106, 2018.

[35] L. E. Margulieux and R. Catrambone. Finding the best types of guidance for constructing self-explanations of subgoals in programming. *Journal of the Learning Sciences*, 28(1):108–151, 2019.

[36] L. E. Margulieux, R. Catrambone, and M. Guzdial. Employing subgoals in computer programming education. *Computer Science Education*, 26(1):44–67, 2016.

[37] L. E. Margulieux, M. Guzdial, and R. Catrambone. Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. In *Proceedings of the ninth annual international conference on International computing education research*, pages 71–78, 2012.

[38] S. Marwan, G. Gao, S. Fisk, T. W. Price, and T. Barnes. Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, pages 194–203, 2020.

[39] S. Marwan, T. W. Price, M. Chi, and T. Barnes. Immediate data-driven positive feedback increases engagement on programming homework for novices. In *Educational Data Mining in Computer Science Education (CSEDM) Workshop @ EDM'20*, 2020.

[40] J. McKendree. Effective feedback content for tutoring complex skills. *Human-computer interaction*, 5(4):381–413, 1990.

[41] A. Mitrovic and S. Ohlsson. Evaluation of a constraint-based tutor for a database language. 1999.

[42] A. Mitrovic, S. Ohlsson, and D. K. Barrow. The effect of positive feedback in a constraint-based intelligent tutoring system. *Computers & Education*, 60(1):264–272, 2013.

[43] D. Palmer. A motivational view of constructivistâinformed teaching. *International Journal of Science Education*, 27(15):1853–1881, 2005.

[44] G. D. Phye and T. Andre. Delayed retention effect: attention, perseveration, or both? *Contemporary Educational Psychology*, 14(2):173–185, 1989.

[45] T. W. Price, Y. Dong, and D. Lipovac. iSnap: Towards Intelligent Tutoring in Novice Programming Environments. In *Proceedings of the ACM Technical Symposium on Computer Science Education*, 2017.

[46] T. W. Price, Z. Liu, V. Catete, and T. Barnes. Factors Influencing Students' Help-Seeking Behavior while Programming with Human and Computer Tutors. In *Proceedings of the International Computing Education Research Conference*, 2017.

[47] T. W. Price, R. Zhi, and T. Barnes. Evaluation of a Data-driven Feedback Algorithm for Open-ended Programming. In *Proceedings of the International Conference on Educational Data Mining*, 2017.

[48] T. W. Price, R. Zhi, and T. Barnes. Hint Generation Under Uncertainty: The Effect of Hint Quality on Help-Seeking Behavior. In *Proceedings of the International Conference on Artificial Intelligence in Education*, 2017.

[49] K. Rivers and K. R. Koedinger. Data-Driven Hint Generation in Vast Solution Spaces: a Self-Improving Python Programming Tutor. *International Journal of Artificial Intelligence in Education*, 27(1):37–64, 2017.

[50] C. Romero and S. Ventura. Educational data mining and learning analytics: An updated survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(3):e1355, 2020.

[51] P. Shabrina, S. Marwan, T. W. Price, M. Chi, and T. Barnes. The impact of data-driven positive programming feedback: When it helps, what happens when it goes wrong, and how students respond. In *Educational Data Mining in Computer Science Education (CSEDM) Workshop @ EDM'20*, 2020.

[52] Y. Shi, Y. Mao, T. Barnes, M. Chi, and T. W. Price. More with less: Exploring how to use deep learning effectively through semi-supervised learning for automatic bug detection in student code. *EDM*, 2021.

[53] Y. Shi, K. Shah, W. Wang, S. Marwan, P. Penmetsa, and T. Price. Toward semi-automatic misconception discovery using code embeddings. In *The 11th International Conference on Learning Analytics Knowledge (LAK 21)*, 2021.

[54] V. J. Shute. Focus on formative feedback. *Review of educational research*, 78(1):153–189, 2008.

[55] D. Sleeman, A. E. Kelly, R. Martinak, R. D. Ward, and J. L. Moore. Studies of diagnosis and remediation with high school algebra students. *Cognitive Science*, 13(4):551–568, 1989.

[56] R. Souza, L. Neves, L. Azevedo, R. Luiz, E. Tady, P. R. Cavalin, and M. Mattoso. Towards a human-in-the-loop library for tracking hyperparameter tuning in deep learning development. In *LADaS@ VLDB*, pages 84–87, 2018.

[57] J. Stamper, T. Barnes, and M. Croy. Enhancing the automatic generation of hints with expert seeding. *International Journal of Artificial Intelligence in Education*, 21(1-2):153–167, 2011.

[58] J. Stamper, T. Barnes, L. Lehmann, and M. Croy. The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems Young*

Researchers Track, pages 71–78, 2008.

[59] D. Toll, A. Wingkvist, and M. Ericsson. Current state and next steps on automated hints for students learning to code. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. IEEE, 2020.

[60] J.-P. Vert, K. Tsuda, and B. Schölkopf. A primer on kernel methods. *Kernel methods in computational biology*, 47:35–70, 2004.

[61] D. Wang, W. Dong, and Y. Zhang. Collective intelligence for smarter neural program synthesis. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pages 98–104. IEEE, 2020.

[62] L. E. Winslow. Programming pedagogy—a psychological overview. *ACM Sigcse Bulletin*, 28(3):17–22, 1996.

[63] D. Xin, L. Ma, J. Liu, S. Macke, S. Song, and A. Parameswaran. Accelerating human-in-the-loop machine learning: challenges and opportunities. In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*, pages 1–4, 2018.

[64] R. Zhi, T. W. Price, N. Lytle, and T. Barnes. Reducing the State Space of Programming Problems through Data-Driven Feature Detection. In *Proceedings of the Educational Data Mining in Computer Science Education Workshop at the International Conference on Educational Data Mining*, 2018.

Automatically classifying student help requests: a multi-year analysis

Zhikai Gao
North Carolina State University
zgao9@ncsu.edu

Collin Lynch
North Carolina State University
cflynch@ncsu.edu

Sarah Heckman
North Carolina State University
sarah_heckman@ncsu.edu

Tiffany Barnes
North Carolina State University
tmbarnes@ncsu.edu

ABSTRACT

As Computer Science has increased in popularity so too have class sizes and demands on faculty to provide support. It is therefore more important than ever for us to identify new ways to triage student questions, identify common problems, target students who need the most help, and better manage instructors' time. By analyzing interaction data from office hours we can identify common patterns, and help to guide future help-seeking. My Digital Hand (MDH) is an online ticketing system that allows students to post help requests, and for instructors to prioritize support and track common issues. In this research, we have collected and analyzed a corpus of student questions from across six semesters of a CS2 with a focus on object-oriented programming course [17]. As part of this work, we grouped the interactions into five categories, analyzed the distribution of help requests, balanced the categories by Synthetic Minority Oversampling Technique (SMOTE) , and trained an automatic classifier based upon LightGBM to automatically classify student requests. We found that over 69% of the questions were unclear or barely specified. We proved the stability of the model across semesters through leave one out cross-validation and the target model achieves an accuracy of 91.8%. Finally, we find that online office hours can provide more help for more students.

Keywords

Office Hour, Computer Science Education Research, Text Analysis, help-seeking request

1. INTRODUCTION

Over the past decade the popularity of CS majors has increased and enrollments have skyrocketed [2]. This has created challenges for instructors with increasing demands for individual support, collaborative learning, and automated guidance [12, 2, 16, 15]. As the size of courses and cohorts have increased, the demand for office hours has begun to exceed the time that instructors and staff have available [12, 13]. To address these needs instructors have adopted a wide range of innovative support models including virtual office hours [10], peer support [7], and ticketing systems for help-seeking interactions [13]. The last approach is exemplified by My Digital Hand (MDH) [21], an online support system for office hours which allows students to queue for office hours, post questions in advance, and record the outcome of interactions. MDH assists students in structuring their help-seeking interactions with teaching staff. It also assists instructors and teaching assistants (TA) in managing their courses, by allowing them to triage student questions and target their effort during office hours to be efficient and meet group and individual needs. MDH also tracks help-seeking and interaction data throughout the whole semester. Using this data, we can identify patterns in students' help requests and automatically classify the questions. One common challenge for help-seeking interaction on large classes arises when many students ask the same or similar questions but must be dealt with separately thus eating up limited instructor time. One approach to address this is to develop automated Q&A systems which can leverage common problems. In order for this to work however, students must provide sufficient information about their problems so that they can receive targeted support.

Our goal is to develop analytical methods to understand what kinds of help students seek during office hours, how they frame their questions to the instructors, and whether or not we can automatically classify questions to support guidance and time management. By analyzing students' help requests across course offerings we can better understand what kinds of challenges the students are facing, and how the teaching staff can better anticipate students' needs and target their limited support. Moreover, by automatically classifying help requests we can help teaching staff to efficiently triage student questions and identify common problems that may be solved with group support or peer assistance. Over the long term we will develop summary

Zhikai Gao, Collin Lynch, Sarah Heckman and Tiffany Barnes "Automatically classifying student help requests: a multi-year analysis". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 81-92. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

statistics which can be used to support instructors in course management, and we will augment our existing ticketing system with support for automatic categorization.

In this paper we will address four specific research questions in the context of a CS2, object-oriented-focused course:

- RQ1: What types of questions do students ask on MDH during office hours and how do they formulate their description?
- RQ2: How can we automatically classify student help requests and tickets?
- RQ3: How robust is our classification model across different offerings?
- RQ4: Compared to regular office hours, does online office hours provide more benefits?

In order to address RQ1 we analyzed our dataset to identify common patterns of student questions and to classify them into five categories. We then address RQ2 and RQ3 by training an automated classifier for student questions with the goal of evaluating its' stability across semesters. Due to the COVID-19 pandemic, all courses are operating online in Fall 2020, which gives us an opportunity to study the advantages and disadvantages of hosting office hour online. Therefore we analyzed and compared the data pattern on Fall 2020(F20) with other regular semesters in RQ4.

1.1 Background

Prior researchers have analyzed student help requests with the goal of understanding student behaviors. Xu and Lynch, for example applied deep learning approaches to classify student question topics in MOOC discussion forums [24]. In that work Xu and Lynch collected student posts from two offerings of a MOOC on Big Data in Education. The authors classified student questions into one of three types (Course Content Question, Technique Question, and Course Logic Question) and developed an automatic classifier using Recurrent Neural Networks to divide questions' into those three categories. While the models were successful within a single offering they Xu and Lynch, found that they did not generalize across offerings. Thus, the system suffered from a cold start problem on each semester.

Vellukunnel et al. in turn collected Piazza posts from CS2 courses offered at two institutions and analyzed the type and distribution of the questions students asked [22]. As part of this work they manually partitioned the questions into five categories and then analyzed the impact of students' question types on their final grades. They concluded that asking constructive questions can help students to develop a better understanding of the course materials and in turn receive better grades. This analysis has informed our own work. However the Piazza platform, unlike MDH, is designed to support interactive discussion and online peer support through the use of threads and replies. By contrast the MDH system is focused on initial help seeking and not on collaborative dialogue. Therefore it is unclear whether our results will align with theirs.

Prior researchers have also studied how instructors manage office hours and how to make face to face support time more efficient and effective. Guzdial, for example, argued that office hours should incorporate diverse teaching techniques including pair programming, peer instruction, and backward design. These approaches, he argued, would potentially work to reduce wait times and support enhanced learning outcomes [8]. In order to provide more convenience for students, Harvard University introduced virtual office hours to an introductory programming course CS50 so that students can interact with teaching staff online [14]. However, they found that those virtual sessions were often inefficient and took more time to address the students' problems. This research is complicated by the fact that students frequently avoid seeking help from teaching staff when they need it [1]. Some of the factors behind this help-avoidance include a lack of trust in the tutor's abilities, inaccessibility of office hours due to timing or other constraints, and a desire for independence in learning [18]. While our research provides some guidance on the design of office hours and the need to reach out to students, the impact of how students frame their help requests has not yet been analyzed extensively. One notable exception is the work of Ren, Krishnamurthi, and Fisler, who designed a survey-based method to help track the students' help-seeking interactions during office hours in programming-based CS courses [20]. While informative, their approach is difficult to generalize as it depends on requiring the teaching staff to complete a detailed form *after* every interaction. In MDH, by contrast, we collect much of the data upfront as an integral part of the process.

2. METHODS

2.1 MDH system

My Digital Hand (MDH) [21], is a ticketing system for office hours that was developed to facilitate large CS courses. Students using MDH request help during office hours by "raising a virtual hand", that is creating a ticket which lists the topic they need help on, describes the issue they are facing, and the steps they have taken to address it. Once the ticket is created it is visible to the teaching staff who can then use it to prioritize interactions or even group students together for help. Once the interaction is complete the teaching staff can close the ticket and describe how the interaction played out. Students are also given the opportunity to evaluate the help received. These feedback questions are configurable and set by instructors at the start of the semester.

This data allows instructors to identify common issues facing students and to track the time it takes for students to receive support from the teaching staff as well as the duration of each help session. A prior analysis of MDH data, Smith et al. found that 5% of students in a course accounted for 50% of office hour time, and that long individual interaction times, representing students who needed long and detailed guidance, served to delay many other short questions [21]. They concluded that a small but critical group of students are reliant on individual tutoring via office hours, while other students who need intermittent help are often unable to obtain support. These findings have motivated our own focus on developing analytical tools which can be used to analyze, prioritize, and manage help requests so that high-demand students do not shut out their peers.

2.2 Data Collection

We analyzed data from seven semesters of a typical second-semester Object-Oriented programming course [17] at a research-intensive public university in the south-eastern United States. Basic descriptive statistics for the dataset are shown in Table 1. Students produced an average of 1477 tickets per semester with higher volumes in the fall semesters due to larger class sizes. The number of tickets also increased year over year due in part to larger class sizes, greater emphasis on tool use by the course instructor, and higher per-capita demand for office hours.

The course is structured as a single lecture section with 12 small-group lab sessions which are held weekly. Over the course of the semester students complete weekly lab assignments, 2-3 individual or team Projects (C-Projects), and 3 separate Guided Projects (G-Projects). The G-Projects are designed to provide a review of prerequisite materials and introduce students to new concepts. The C-Projects are generally structured as two sub-assignments, one focused on design and system testing, and the other on implementation and unit testing. Students manage their code via the GitHub platform with integrated support for the Jenkins automation testing server. When students commit code they receive automated testing results from instructor-authored test cases as well as test cases that they supplied. The students use feedback from test failures to guide their work and their help-seeking.[6]

In Fall 2020, this course was moved fully online due to the pandemic. All office hours were hosted through zoom meeting where students can share their screen with the teaching staff to show their code or any problems.

Table 1: Number of tickets and students for each semester (F= Fall, S=Spring)

	F17	S18	F18	S19	F19	S20	F20
tickets	1146	609	1224	860	1650	1401	3452
students	208	157	259	174	256	191	303

The interaction data is the most important for our current analysis. The format of the interaction records, along with selected examples is shown in Table 2. For this analysis each ticket consists of three major parts: the participants, time and duration of interaction, and the context.

2.3 RQ1: Categorization of Questions

Our primary focus in RQ1 is to identify the types of questions the students are asking and to understand how they describe their work. MDH allows students to frame their question topic or description in any way that they wish. The platform does not provide a list of suggested topics or mandate content beyond the basic text. This, in turn, lead students to vary widely in the descriptions and content that they provide. We therefore studied two features of the questions with the goal of supporting classification, the students' topic, as contained in the `"I'm working on"` field. And the longer problem description, as stated in the `"my problem is"` field.

2.3.1 *Classified by Topic*

Table 2: Attributes of the interaction data

Attributes	Content Explain	Example
interaction id	Id for each ticket	30072
student id	Id for the student who raised this ticket	1950
teacher id	Id for teacher who deal with the ticket	20810
time raised hand	Timestamp for each tickets that are asked	2019-03-08 19:34:09
time interaction began	Timestamp for each tickets began	2019-03-08 19:38:39
time interaction ended	Timestamp for each tickets ended	2019-03-08 20:01:02
I'm working on	Topic for the question of each ticket	Program1Part1
my problem is	Detail statement for the question of each ticket	Null Pointer on TS test
I've tried	The solution the student tried before they raised the tickets	Debugging

Rapidly identifying, or even anticipating, students' question topics would allow teaching staff to anticipate the kinds of issues they should be prepared for and may also allow them to set up mini-groups within office hours to deal with problems assignment by assignment, or to separate code questions from conceptual ones. We therefore performed a manual analysis of the topics in our study dataset over all semesters with the goal of determining how students label their topics, and whether it is possible to either anticipate or sort their posts as they come in.

Our preliminary analysis showed that in most cases the students simply entered the name of their current assignment or an abbreviation of it and provided no other details. Moreover, due to the structure of the course deadlines almost every help request in a given session was focused on the same assignment. In the newest version of MDH, the question is now a check box and the instructor can set the assignments. As a consequence we decided to omit this from our classification task and focus on the types of help being sought.

2.3.2 *Classified by Description*

In the description section ("my problem is"), the students can provide a rich summary of their problem including a text description, bug reports, or even code snippets. If it is possible to automatically classify student posts then we can use that approach to triage student questions as they come in, perhaps separating long questions from short. We therefore performed a manual analysis of the description content as well with the goal of identifying useful categories of posts. We also sought to examine how complex the problem descriptions were. In our prior discussions with the teaching staff they reported that many students provide too little information in the description (e.g. a single word such as "Errors"), provide too much (e.g. a full execution dump and error log), or they simply type gibberish with the simple

goal of securing a place in line. All of these strategies are problematic either because they provide too little information to effectively triage posts, or because the dump is too complex and likely out of date before the student reaches the head of the line. In our analysis we analyzed both the length and structure of the students' submissions as well with the goal of understanding whether we can provide automatic scaffolding for useful posts, and automatic triage of the submissions.

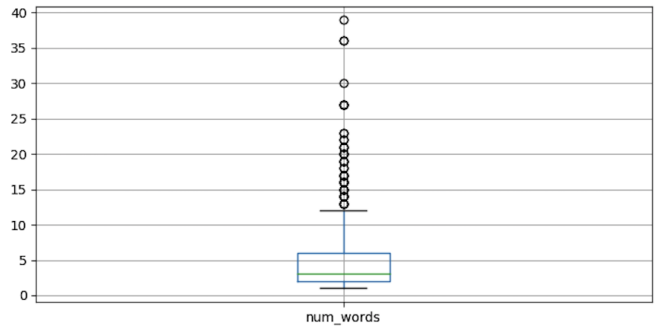


Figure 1: Distribution of the number of words in the questions' description

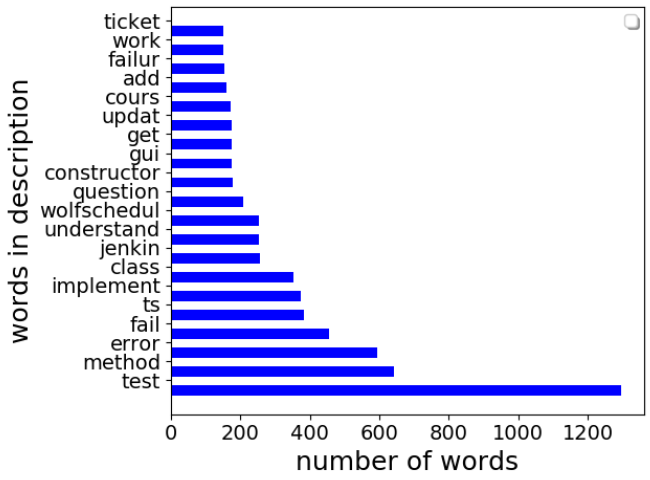


Figure 2: Top 20 Popular words in question description

We examined the length, content, and complexity of the students' problem description including the specific `my problem is` prompt. We also grouped common words to find keywords that are associated with specific types of submissions. We then used this information to inform our understanding of the students' posting behavior and to inform the design of the posting categories. This preliminary analysis was consistent with the experience described by the teaching staff. Figure 1 shows that the average description was less than five words long. When reviewing those short posts we noted that many students preferred to use keywords to indicate their question topics and problems rather than spelling their issues out. For example, when encountering an error in implementing an `add()` function, they often put "add()" as the problem description assuming that the method name, together with the assignment information, provided enough context for the help request.

This led us to focus on the specific terms that students use in their problem description. In this analysis, we grouped words by stemming and ignored stopwords to focus on the primary content information. The top 20 words are shown in Figure 2, top among them being `test`. This is consistent with the design of the assignments where students were provided with tests and required to develop their own. It is also consistent with the teaching staff's observation that many students focus on the tests as a guide for their progress and for where they need help. Upon closer examination of questions using this word we found that most interactions were focused on failed test cases; a typical description for a question of this type was "2 test case fail". It would be difficult for instructors to interpret these without reviewing the code and the test results in more detail but such a review takes time. Another closely-related word that was common in this dataset is `error` which was used primarily when students encounter bugs or other failures. In these cases in particular, the teaching staff noted that some students would simply paste the crash report into the question with little other context. This kind of behavior is rare in the data but was also useful for instructors, we therefore used it as an additional factor.

Based upon this preliminary analysis we defined five categories of help requests based upon the problem descriptions. These categories are shown in Table 3. We then labeled all interactions related to the problem manually. For each question, we also ranked the clarity or comprehensibility of student questions based upon the description provided. As we discuss below, most of the questions provided *insufficient* information to diagnose the problem. However as Figure 1 shows, some students did elaborate on their problem thoroughly as measured by the number of words in the problem description.

2.4 Labeling Process

2.4.1 Code book

To investigate the distribution of the above five categories in our data, we first need to set up a standard to categorize our data and apply it. All seven semesters' data was labeled by one researcher by the following rules:

- Check if there is any text that is clearly an error message copied from the compiler or a test failure. If so, label it as Copied Error. Notice that if the student describes the error message in their own words, then it should also be classified as Sufficient.
- Check if there is any text indicating that this is a test problem, no matter if the description gives you the detail of their test error or not. If you are sure that it is a test problem, label it as Test. If it also qualifies as Copied Error, classified as Copied Error
- If the text does not provide any information about their question and you cannot understand or deduce anything that related to their question, classify as Useless

Table 3: Explanation and example for all five categories we developed

Category	Explanation	Example description(my problem is)
Useless	The description contain nothing related to the question	I would like to check of it
Insufficient	The description contains partial information about the student question, but not enough for instructors to understand the details.	TicketManager getInstanceOf
Sufficient	Contains enough detail on the question for instructors to understand. Usually a very clear sentence.	CourseRecordIOTest, I seem to be failing reading the files, at the moments it is testing the size of the ArrayLists, but I am passing writing the files
Copied Error	Contains a copied error from the compiler.	I got this error: TypeError: barh() got multiple values for argument 'width'
Test	Test case fail related problem	2 test cases failed

- If the text does provide what or where the problem is, but not enough for you to fully understand or identify what is their question, marked as Insufficient
- If the text only contains one or multiple words of the method name associated with a problem, it can help to localize the problem but provides no additional details. It should be classified as Insufficient
- If the text is in a form of "I don't understand xxxx" without further explanation of which part they do not understand or other details, classify as Insufficient
- If the text is in a form of "I don't understand xxxx" with some further explanation, classify as Sufficient
- If the text tells you what their question or describes how they encounter this problem, classify as Sufficient

2.4.2 Inter rater reliability

After all data was labeled, we randomly generated a subset of 150 unique questions(30 for each category) and sent it to another researcher to rate. In this subset, we reveal the label of 10 questions for each category as example data and the rater classifies the remaining questions based on those example data and the code book. Then we compare the result with the original labels and calculate Kappa to represent the inter rater reliability. Kappa[4] is widely applied for measuring the agreement between two coders that accounts for chance agreement. Generally a score higher than 0.8 is considered acceptable. In our cases, the final unweighted Kappa value is 0.815 which is acceptable.

2.5 RQ2: Modeling

In addressing RQ2 we drew on our basic categorization developed in RQ1 to train automatic classifiers that can triage posts by topic and content. We used the first six semester data as training data to train our model and the last semester (F20) as the testing set to evaluate our model. To train our classification model, we first extracted training features from the problem descriptions across our dataset. The features included content features such as the keywords described above as well as meta-text features such as length, the number of stop-words (as a general proxy for specificity), the punctuation, and the character case. These meta-text features have the advantage that they are easy to extract automatically and can therefore be used for automated triage. Length, for example, is a suitable proxy for completeness and coherence while punctuation and case shifting are common in error messages. The full list of these features is shown in table 4.

We represented the text features as a tf-idf [19] matrix and basic word count matrix over the content. The word count matrix is simply a 2D Array which describe how many times each term appears in each question text. The tf–idf matrix is the product of two statistics, term frequency and inverse document frequency. The term frequency uses the raw count of a term in a text. The inverse document frequency is a measure of how much information the word provides. Some common words like "is" or "that" do not provide much information but they do usually have a high term frequency. Those words should have less inverse document frequency(idf). We can calculate the value as:

$$idf(t) = ln(\frac{Total\,number\,of\,documents}{Number\,of\,documents\,with\,term\,t\,in\,it}) \quad (1)$$

In our preliminary analysis we found that the matrices performed poorly in classification due to the fact that both were extremely sparse. We therefore opted to compress them so that they can be compatible with the dense feature approaches. To that end we built a Naive Bayes model [9] using the tf-idf sparse features and then use the predictions features. From this model we generated five shallow prediction features which correspond to the probability that the question belongs to each category. We followed this same approach with the word count vector and used those features as probabilities. The final list of extracted features is shown in Table 5.

2.5.1 Model Training

We trained our classification models using LightGBM [11], a Gradient Boosting Decision Tree (GBDT) algorithm provided by Microsoft. GBDT is an ensemble model of decision trees trained in sequence. In each iteration, GBDT learns the decision trees by fitting the negative gradients

Table 4: Text meta features list

Feature name	Explanation	value example
length	number of words in the problem description	12
character	number of characters in the problem description	12
stop words	number of stop words in the problem description	2
punctuation	number of punctuation in the problem description	0
uppercase	number of uppercase words in the problem description	0

Table 5: Text content features lists

Feature name	Explanation	value example
prob-tf-useless	the probability of this tickets belong to category "useless" using tf-idf	0.75
prob-tf-ins	the probability of this tickets belong to category "insufficient" using tf-idf	0.82
prob-tf-suf	the probability of this tickets belong to category "sufficient" using tf-idf	0.77
prob-tf-error	the probability of this tickets belong to category "copied error" using tf-idf	0.99
prob-tf-test	the probability of this tickets belong to category "test" using tf-idf	0.65
prob-cnt-useless	the probability of this tickets belong to category "useless"using using common word count	0.75
prob-cnt-ins	the probability of this tickets belong to category "insufficient" using common word count	0.82
prob-cnt-suf	the probability of this tickets belong to category "sufficient"	0.77
prob-cnt-error	the probability of this tickets belong to category "copied error" common word count	0.99
prob-cnt-test	the probability of this tickets belong to category "test" common word count	0.65

(also known as residual errors). To reduce the complexity of GBDT, LightGBM utilize two novel techniques to improve the algorithm: Gradient-based One-Side Sampling and Exclusive Feature Bundling. This method also utilizes a Leaf-wise Tree Growth algorithm to optimize the accuracy of the model and it applies a max depth of the trees to overcome the over-fitting problem that it might cause. Further, it optimizes the speed of training by calculating the gain for each split and uses histogram subtraction. LightGBM is known for its outstanding performance and relatively good speed. Thus, many researches applied this method to machine learning tasks.

The implementation code for LightGBM was provided by Microsoft[11] in 2013 and we are utilizing its Python library for modeling process. We applied features and the label of training data by LightGBM to train a model, and fit that model on the testing data to predict each question in those data. By calculating the accuracy of the prediction, we can evaluate the performance of this model. We ran a series of 20 preliminary experiments to explore the space of parameters before we settled on the values listed in Table 6. A list of crucial parameters people generally need to tune to improve classification model performance is also in Table 6. Since our goal is to achieve better Accuracy, we will tuning toward larger max_bin, smaller learning_rate with larger num_iterations, larger num_leaves and larger max_depth each experiment until the accuracy is not improving.

2.5.2 SMOTE

During the modeling process, another issue we faced is that the categories are highly imbalanced. Over half of the questions are in the *Insufficient* category and the *Copied Error* category contained fewer than one percent of questions. To address the problem, we applied SMOTE method which over-samples examples in the minority class. SMOTE [5], first selects one minority class instance at random, create a synthetic instance by choosing one of the k nearest neighbors at random and connecting those two instance to form a line segment in the feature space. We applied this method with k=5 and oversampling the data to generate the training datasets and testing datasets for further model training.

2.6 RQ3: Model stability over semesters

For a trained model to be useful however, it must be stable across semesters or else we suffer from a *cold-start problem* [3]. In order to assess the model stability we ran a series of experiments where we assessed the relative utility of the models by applying a leave-one-out validation strategy on a semester-by-semester basis. Showing that all models perform at a comparable level provides a strong indication that the models themselves are consistent and useful, even early in the semester.

2.7 RQ4: Online Office hour analysis

In Fall 2020, all the office hours were held online, which provided valuable data about online office hours interactions. We are very curious to analyze and see whether the students behavior changed with the move to online office hours and if we should keep some online office hours sessions once we resume in-person instruction.

We first analyzed whether the online session attracted more students to seek help during office hours. For an in-person session, students need to physically find the teaching staff in the office and physically stay in line. For online sessions, students only need to click the link to join the meeting with teaching staff. With online office hours, the friction of physically going to a campus location has been removed. However, online office hours have additional overhead in creating a connection between parties and transitioning between students. To better understand online office hour help-seeking, we calculated the average number of tickets per student and the percentage of students who used office hour in each semester and compared earlier semesters with in-person office hours to the Fall 2020 semester with online office hours.

Table 6: LightGBM common training parameters and the final optimal value after tuning our model

parameter	meaning	final optimal value
num_leaves	number of leaves in one tree	1000
max_depth	Specify the max depth to which tree will grow.	10
max_bin	max number of bins to bucket the feature values.	150
learning_rate	learning rate of gradient boost	0.1
num_iterations	number of boosting iterations to be performed	32
num_class	number of classes. used only for multi-class classification	5

Table 7: Distribution of labeled questions on those five categories in all seven semesters

Useless	Insufficient	Sufficient	Copied Error	Test
3.01%	69.02%	12.04%	0.10%	15.83%

Table 8: Average interaction time(in minutes) and standard deviation of each semester

	Useless	Insufficient	Sufficient
AVG	19.7	21.9	18.3
STD	125.3	237.0	103.6
	Copied Error	Test	
AVG	11.5	24.8	
STD	67.5	208.2	

However, online office hours could have an impact on the efficiency of communication. When teaching staff and students meet online, it creates more challenges for teaching staff to indicate problems to the students and to help explain why their code is failing. Screensharing allows the staff to view the students' work, but physical interactions like pointing to a portion of the screen to indicate which button to click is lost. The teaching staff member needs to verbally describe the debugging process and ask students to follow it. Therefore, we calculated the interaction time and the wait time for each ticket. The we compare the distribution of interaction time and wait time of F20 tickets with the rest of tickets. Additional overhead is incurred when connecting to a meeting. There is a lag when a student joins a meeting for their audio to set up to start the conversation.

3. RESULTS

3.1 RQ1: Categorization Results

Table 7 shows the distribution of question categories across our dataset. As the figure shows, the most common category is *Insufficient* which occupies over 69 percent of the questions. The *Test* category coming next at 15 percent. Approximately 12 percent of the questions belong to the *Sufficient* category while 3 percent were rated as *Useless*. Surprisingly, despite comments from the teaching staff, the least common category was *Copied Error* with at most 10-15 questions per semester falling into this group. As our results show, the students tended to use the system primarily as a way of getting in line and typically provided little useful information for the teaching staff. These results also highlight the significance of testing tasks for the assignments and for students' help-seeking given the high proportion of help tickets that are triggered by them.

To assess the stability of these results we also examined the frequencies within each semester. Figure 3 shows this breakdown. We found that the relative distribution is generally similar across semesters while the absolute percentages vary. In more recent semesters the students have authored more *Sufficient* tickets than in prior years suggesting that there has been greater effort by the instructional staff to encourage good communication. Yet the persistence of the other ticket type suggests that automatic classification and triage remain an important feature.

Table 8 shows the average and standard deviation of interaction time (the difference between when the interaction began and it was closed) of each category across the semesters. For this calculation we did not consider tickets with an interaction time less than 10 seconds in length or which were longer than one hour. Our discussion with teaching staff and the instructors showed that the former were cases that were never seen as the student set a placeholder but fixed their problem before their turn came up or changed their mind, while the latter represents cases where the teaching staff offered help but did not close the ticket, often until well after the tutoring session was over. The *Useless*, *Insufficient* and *Sufficient* categories averaged around twenty minutes in length with no meaningful difference in their interaction times. The *Copied Error* category was slightly shorter on average which may reflect the specificity of the students' problems while the *Test* category had a slightly longer average interaction time. This may indicate that this kind of question is more complex or more substantive relative to the others. Overall these results indicate that the amount of information provided does not necessarily affect the speed with which the issue can be addressed.

3.2 RQ2: Modeling Results

To evaluate the performance of our model, we trained the model using the first six semesters' data and tested it on Fall 20 data.The training dataset applied SMOTE method to oversampling the minority categories and result in each category having the same amount(5037) of questions in training dataset. The model achieved an overall accuracy of 91.8%. We then conducted a more detailed analysis of the performance for precision, recall, and F-score on each question type. The results are in Table 9. As our results show the model is relatively balanced across the categories with the exception of the Test category which had substantially higher precision and lower recall. This indicates that it was far more likely for other categories to be erroneously classi-

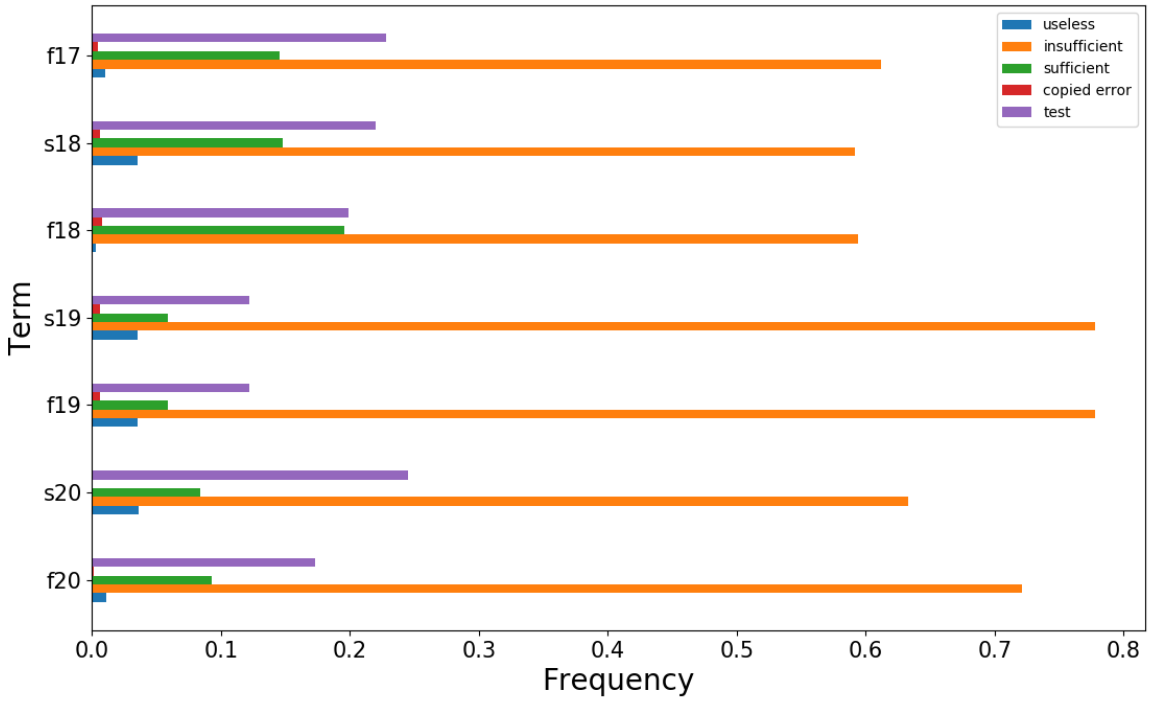


Figure 3: Frequency of each category for each semester

fied as *Tests* when the student submitted other information. One comment that triggered this error was: "Jenkins errors".

Table 9: Precision, Recall and F-measure for each category

	Useless	Insufficient	Sufficient	Test	AVG
Precision	0.902	0.915	0.913	0.967	0.922
Recall	0.901	0.903	0.905	0.877	0.901
F-measure	0.896	0.894	0.901	0.914	0.901

3.3 RQ3:Leave One out Results

Having shown that the model is relatively balanced across categories we then analyzed the relative accuracy of the model on each semester. The results are shown in Table 10. With the exception of Fall 2019, the model achieved an accuracy of at least 0.91 across each semester. Fall 2019 was the largest and busiest semester in our training dataset which may indicate that students were more diverse in their habits or posting behavior but even still we achieved an accuracy of 0.899. In light of these results we believe that our modeling method is sufficiently stable to assist in processing unseen semesters without a cold-start problem.

Table 10: Leave one out accuracy result

left-out semester	F17	S18	F18	S19	F19	S20
accuracy	0.913	0.915	0.908	0.907	0.899	0.912

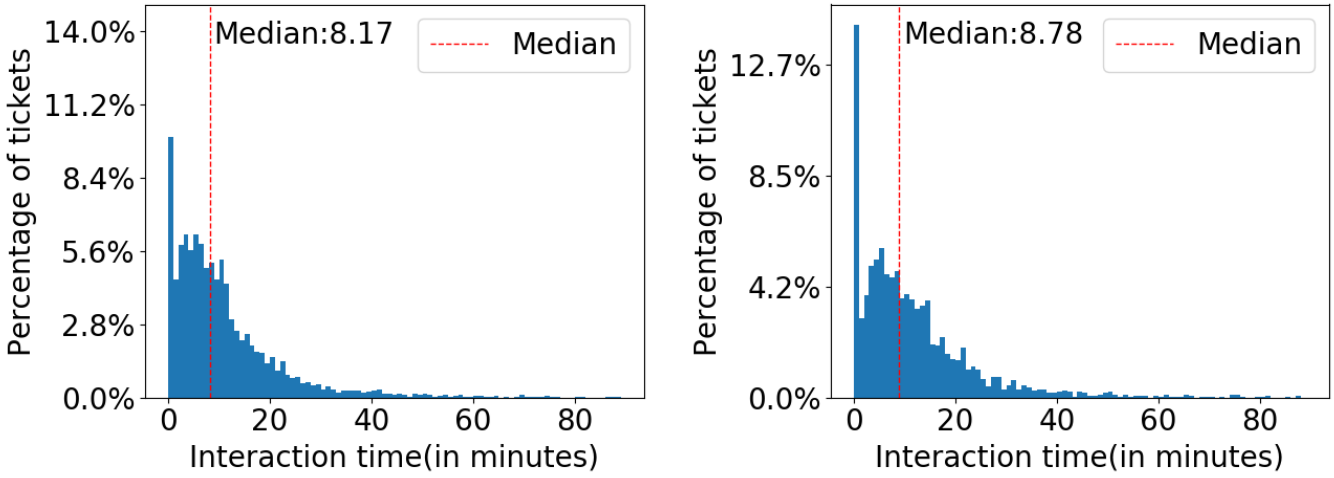
3.4 RQ4: Online Office Hour Analysis Results

Table 11 shows the various summary measures associated with office hours interactions for the semesters studied. The Fall 2020 semester had the highest number of average tickets per student and the largest percentages of students utilizing office hours. Additionally, the average tickets per student in Fall 2020 is nearly twice the average tickets per students in Fall 2019. This suggests that online office hours supports increased student participation in office hours interactions. However, this increment could be explained by other factors. The data shows that in general more students take advantage of the help each year. This is consistent with the increasing class sizes but it is important to note that there are other patterns as well such as regular dips in each spring. Overall it serves to highlight the need for better course management. Secondly, since the course lecture also holds online in F20, the increase of office hour usage supports a general expectation that students are facing additional challenges with online classes however we still believe that online office hours are a practical means to minimize the cost of help-seeking and thus encourage more students.

The distribution of interaction times shown in Figure 4 indicates that the teaching staff generally took slightly longer to support students in Fall 2020 than other semesters. The median interaction time of Fall 2020 is 8.78 minutes while other semesters are 8.17 minutes. We believe that this is caused by the inconvenience of remote instruction and debugging. Additionally, the high percentages of interactions within one minute in all semesters are usually caused by teaching staff forgetting to open the tickets when the interaction begins. In Fall 2020, the percentage of short tickets was much higher suggesting the teaching staff were more likely to make such mistakes because they are working with both MDH and the online interaction tool. After notify the student, the teaching staff has to waited in the zoom until

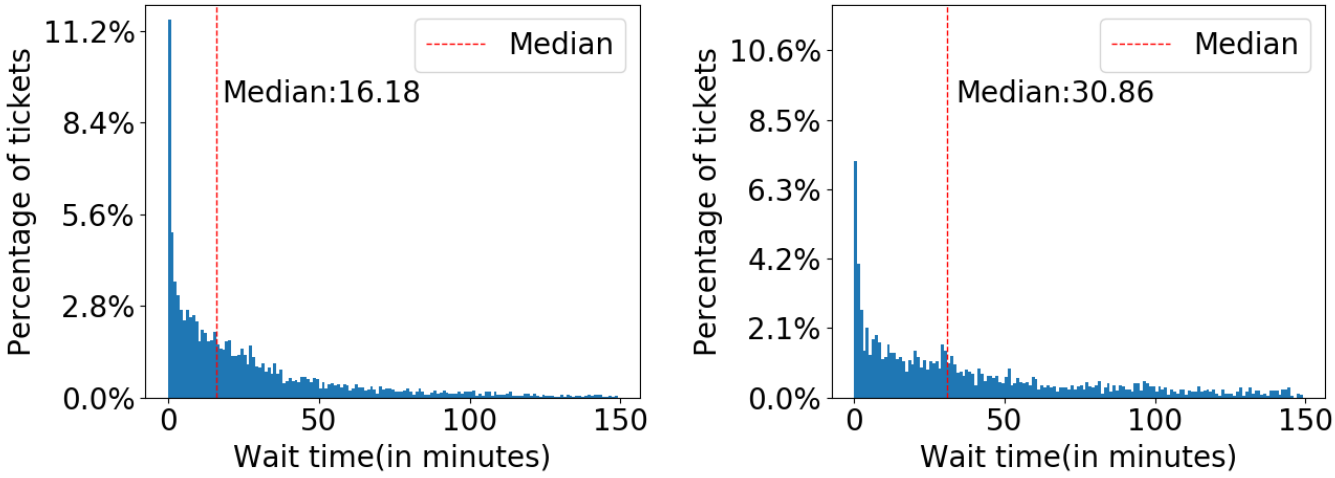
Table 11: Statistic Analysis for each semester

	F17	S18	F18	S19	F19	S20	F20
total tickets	1146	609	1224	860	1650	1401	3452
total students	208	157	259	174	256	191	303
students use office hour	104	63	141	84	158	108	209
average tickets per student	5.51	3.88	4.73	4.94	6.44	7.34	11.40
percentage of students using office hour	50.0%	40.1%	54.4%	48.3%	61.7%	56.5%	69.0%



(a) (b)

Figure 4: Histogram of Interaction time (time difference between open time and close time) for tickets in (a) Regular semesters and (b)Fall 20



(a) (b)

Figure 5: Histogram of wait time (time difference between open time and raised time) for tickets in (a) Regular semesters and (b)Fall 20

the student actually join the zoom meeting to start the interaction. It is very common for the teaching staff just start the interaction in zoom without open ticket in MDH. The wait time in Fall 2020 is much longer than regular semesters, as shown in Figure 5. While we had 37 hours of regular office hour time each week for 17 TAs and 2 instructors, the increased demand in office hour help-seeking did have impact on student wait time and throughput. We additionally observed an increase in the number of canceled tickets. While online office hours lowers barriers for student attendance, additional resources to support demand are needed to ensure timely support. As we transition back to in-person instruction, the course instructor will continue to offer some online office hours to support access to help-seeking.

4. CONCLUSIONS

The results of our analysis in RQ1 show that students' use of the MDH platform does vary substantially from the developers' intentions. Far from using it to write complex help requests many do use it merely to reserve a place in line. Of the five categories, our labeling showed that most of the help tickets submitted lack sufficient detail to be clear what the student is asking about while those that do provide detail are most commonly focused on test cases which constitute a major feature of the class. Contrary to our initial expectations, the students rarely use the system to enter specific error messages, even if they have them. Thus the teaching staff have relatively little to go on when triaging questions. Clearly the proportion of useful information in the tickets increased in more recent years of the course but insufficient detail remains the most common feature. Despite this however, our results also show that there are clear categories of use that we can build upon to assist teaching staff. And our results show that it may be possible to extend the system with minimal automated interventions such as detectors for word counts or grammar that can be used to scaffold, or simply enforce, good posting behavior.

Informed by our analysis, we were able to address our modeling questions, RQ2 and RQ3 by developing accurate and robust classification models that achieved an overall accuracy of 92.6% and individual accuracy of 0.899% to 0.91% . Moreover, the results are robust on a per-category basis. While these results are not perfect, they show that we have the potential to use models of this type for effective triage of student questions as well as to provide scaffolding and immediate guidance for students as they author help tickets. While such guidance has not been evaluated for its' educational impact prior work on self-explanation (e.g. [23]) leads us to conclude that it may help students to diagnose their own challenges.

For RQ4, the comparison between an online session semester and regular semester shows both the strength and weakness of online office hours. The advantages of hosting office hours online is that it can encourage students to utilize the help-seeking resources; However, the large amount of help-seeking requests can be overloaded for teaching staff and the remote debugging through screen sharing is clearly less efficient than face-to-face interactions.

5. LIMITATIONS

There are several limitations to our work that must be acknowledged. While our results span semesters, they are still taken from a single course with a single instructor. As a consequence our results are necessarily dependant on the training that students have received and it is not yet clear whether this stability will be apparent in models created from interaction data for other courses, particularly those that are not as large, do not use the same assignment structure, or rely so heavily on tests.

Additionally, for our analysis toward online office hour, we did not consider the influence of teaching lectures online could raise more challenges for students and thus increase the usage of office hour. Our conclusion of online office hour encourage students to seek help is based on the assumption that there is no significant difference of academic difficulty between F20 and other semesters.

6. FUTURE WORK

This research can support future instructors in course management and the automatic categorization for MDH system. We therefore plan to address these limitations, expand our dataset, and build upon the models that we have obtained. First, we plan to conduct a more robust process of tagging and classifying our tickets with the goal of assessing the stability of our categories with other evaluators and of identifying other important ways of grouping the tickets themselves.

Second, we will extend My Digital Hand to take advantage of these trained models in supporting both the students and instructors. We will support instructors by providing automatic triage approaches that can help to guide their planning. And we will use automated guidance to prompt students to produce better tickets in the first place.

Third, we also plan to investigate other aspects of the office hours that are captured in the MDH data. These include: whether students in the same office hours post similar tickets, thus highlighting the potential of peer feedback; and the presence or absence of serial ticketers; that is students who keep multiple follow-up tickets going to monopolize support. We plan to build models for these features with the goal of understanding how help time is being used and by extension how to better coordinate limited support.

Finally, we plan to apply our models to provide automated scaffolding for students when they provide insufficient comments or errors. Specifically, we will integrate this model to the MDH system and every time a student raise a hand, we will use our model to predict the question category. If their description is insufficient or useless, then we can immediately notify them to revise it. This will help students to better frame their questions, and it will help the teaching staff can be better prepared to answer the students' question. This initial filter can be followed by additional models to suggest debugging steps or common answers based upon their revised question.

7. ACKNOWLEDGEMENTS

This research was supported by NSF #1821475 "Concert: Coordinating Educational Interactions for Student Engagement" Collin F. Lynch, Tiffany Barnes, and Sarah Heckman (Co-PIs).

References

[1] Vincent Aleven and Kenneth R Koedinger. "Limitations of student control: Do students know when they need help?" In: *International conference on intelligent tutoring systems*. Springer. 2000, pp. 292–303.

[2] Computing Research Association et al. *Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006.(2017)*. 2017.

[3] Tiffany Barnes and John C. Stamper. "Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data". In: *Intelligent Tutoring Systems, 9th International Conference, ITS 2008, Montreal, Canada, June 23-27, 2008, Proceedings*. Ed. by Beverly Park Woolf et al. Vol. 5091. Lecture Notes in Computer Science. Springer, 2008, pp. 373–382. DOI: 10.1007/978-3-540-69132-7_41. URL: https://doi.org/10.1007/978-3-540-69132-7%5C_41.

[4] Kenneth J. Berry and Jr. Paul W. Mielke. "A Generalization of Cohen's Kappa Agreement Measure to Interval Measurement and Multiple Raters". In: *Educational and Psychological Measurement* 48.4 (1988), pp. 921–933. DOI: 10.1177/0013164488484007.

[5] N. V. Chawla et al. "SMOTE: Synthetic Minority Oversampling Technique". In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953.

[6] *csc 216 software development fundamentals ,Engineering Online, NC state university*. June 2020. URL: https://www.engineeringonline.ncsu.edu/course/csc-216-software-development-fundamentals/.

[7] Zhijiang Dong, Cen Li, and Roland H. Untch. "Build Peer Support Network for CS2 Students". In: *Proceedings of the 49th Annual Southeast Regional Conference*. ACM-SE '11. Kennesaw, Georgia: Association for Computing Machinery, 2011, pp. 42–47. ISBN: 9781450306867. DOI: 10.1145/2016039.2016058.

[8] Mark Guzdial. "Cutting the Wait for CS Advice". In: *Commun. ACM* 62.8 (July 2019), pp. 12–13. ISSN: 0001-0782. DOI: 10.1145/3339456.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009. URL: http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

[10] Jeremy Johnson et al. "Virtual Office Hours Using TechTalk, a Web-Based Mathematical Collaboration Tool". In: *Proceedings of the 6th Annual Conference on the Teaching of Computing and the 3rd Annual Conference on Integrating Technology into Computer Science Education: Changing the Delivery of Computer Science Education*. ITiCSE '98. Dublin City Univ., Ireland: Association for Computing Machinery, 1998, pp. 130–133. ISBN: 1581130007. DOI: 10.1145/282991.283094.

[11] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3146–3154. URL: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf.

[12] Kevin Lin. "A Berkeley View of Teaching CS at Scale". In: *arXiv preprint arXiv:2005.07081* (2020).

[13] Tommy MacWilliam and David J. Malan. "Scaling Office Hours: Managing Live Q&A in Large Courses". In: *J. Comput. Sci. Coll.* 28.3 (Jan. 2013), pp. 94–101. ISSN: 1937-4771.

[14] David J. Malan. "Virtualizing Office Hours in CS 50". In: *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE '09. Paris, France: Association for Computing Machinery, 2009, pp. 303–307. ISBN: 9781605583815. DOI: 10.1145/1562877.1562969.

[15] Engineering National Academies of Sciences, Medicine, et al. *Assessing and responding to the growth of computer science undergraduate enrollments*. National Academies Press, 2018.

[16] E. Patitsas, M. Craig, and S. Easterbrook. "How CS departments are managing the enrolment boom: Troubling implications for diversity". In: *2016 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. 2016, pp. 1–2.

[17] Leo Porter et al. "Developing Course-Level Learning Goals for Basic Data Structures in CS2". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE '18. Baltimore, Maryland, USA: Association for Computing Machinery, 2018, pp. 858–863. ISBN: 9781450351034. DOI: 10.1145/3159450.3159457.

[18] Thomas W. Price et al. "Factors Influencing Students' Help-Seeking Behavior While Programming with Human and Computer Tutors". In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER '17. Tacoma, Washington, USA: Association for Computing Machinery, 2017, pp. 127–135. ISBN: 9781450349680. DOI: 10.1145/3105726.3106179.

[19] J. Ramos. "Using TF-IDF to Determine Word Relevance in Document Queries". In: 2003.

[20] Yanyan Ren, Shriram Krishnamurthi, and Kathi Fisler. "What Help Do Students Seek in TA Office Hours?" In: *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ICER '19. Toronto ON, Canada: Association for Computing Machinery, 2019, pp. 41–49. ISBN: 9781450361859. DOI: 10.1145/3291279.3339418.

[21] Aaron J. Smith et al. "My Digital Hand: A Tool for Scaling Up One-to-One Peer Teaching in Support of Computer Science Learning". In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '17. Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 549–554. ISBN: 9781450346986. DOI: 10.1145/3017680.3017800.

[22] Mickey Vellukunnel et al. "Deconstructing the Discussion Forum: Student Questions and Computer Science Learning". In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '17. Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 603–608. ISBN: 9781450346986. DOI: 10.1145/3017680.3017745.

[23] Arto Vihavainen, Craig S. Miller, and Amber Settle. "Benefits of Self-Explanation in Introductory Programming". In: SIGCSE '15. Kansas City, Missouri, USA: Association for Computing Machinery, 2015, pp. 284–289. ISBN: 9781450329668. DOI: 10.1145/2676723.2677260.

[24] Yiqiao Xu and Collin F Lynch. "What do you want? Applying deep learning models to detect question topics in MOOC forum posts?" In:

Early Prediction of Museum Visitor Engagement with Multimodal Adversarial Domain Adaptation

Nathan Henderson, Wookhee Min, Andrew Emerson, Jonathan Rowe, Seung Lee, James Minogue, and James Lester

North Carolina State University
Raleigh, North Carolina, 27695, USA
{nlhender, wmin, ajemerso, jprowe, sylee, james_minogue, lester}@ncsu.edu

ABSTRACT

Recent years have seen significant interest in multimodal frameworks for modeling learner engagement in educational settings. Multimodal frameworks hold particular promise for predicting visitor engagement in interactive science museum exhibits. Multimodal models often utilize video data to capture learner behavior, but video cameras are not always feasible, or even desirable, to use in museums. To address this issue while still harnessing the predictive capacities of multimodal models, we investigate adversarial discriminative domain adaptation for generating modality-invariant representations of both unimodal and multimodal data captured from museum visitors as they engage with interactive science museum exhibits. This approach enables the use of pre-trained multimodal visitor engagement models in circumstances where multimodal instrumentation is not available. We evaluate the visitor engagement models in terms of early prediction performance using exhibit interaction and facial expression data captured during visitor interactions with a science museum exhibit for environmental sustainability. Through the use of modality-invariant data representations generated by the adversarial discriminative domain adaptation framework, we find that pre-trained multimodal models achieve competitive predictive performance on interaction-only data compared to models evaluated using complete multimodal data. The multimodal framework outperforms unimodal and non-adapted baseline approaches during early intervals of exhibit interactions as well as entire interaction sequences.

Keywords

Museum learning, visitor engagement, adversarial domain adaptation, early prediction, multimodal learning analytics.

1. INTRODUCTION

Visitor engagement is critical in museum learning [21]. Engagement defines how visitors experience museums, including how they move between exhibits, form and express interests, and acquire knowledge and understanding. Developing computational models of museum visitor engagement holds significant promise for identifying salient patterns of visitor behavior as well as providing insight into how specific exhibits can be designed to enhance engagement. For example, visitor analytics show potential for enabling adaptive learning experiences tailored to the preferences and tendencies of the visitors, leading to highly engaged interactions with the exhibit. Visitor interactions with museum exhibits are inherently *multimodal*. Visitor engagement manifests through a variety of behaviors such as facial expression, touch, eye gaze, and body posture. As such, multimodal learning analytics can model museum visitor engagement by capturing and analyzing visitor behavior from several different perspectives [2, 16]. Multimodal models of learner engagement have been shown to be effective in a range of environments, including laboratory [8, 22] and classroom settings [1, 6, 7]. More recently, multimodal learning analytics have been the subject of growing attention in informal education settings, such as museums [16, 20], but this line of investigation is still in its nascent stages.

Given the multimodal nature of visitor interactions in museums, the use of multichannel data provides important benefits for modeling visitor engagement. In particular, multimodal models can be used to predict visitor engagement early during a visitor's interaction with an exhibit [16]. This shows promise for enabling visitor-adaptive technologies that provide adaptive support for fostering engaged learning experiences with an exhibit or for notifying museum educators to inform decisions about staffing the museum floor. In predictive modeling, it is important that the multimodal visitor engagement models be evaluated in terms of both predictive accuracy and the minimum amount of time that the models require to achieve robust predictive performance.

Multimodal modeling of visitor engagement in museums also poses significant challenges. Interactions with exhibits are highly variable due to the free-choice nature of museum learning [12, 25, 28]. Additionally, multimodal frameworks often utilize physical sensors (e.g., video cameras, motion sensors, eye trackers), which introduce questions about scalability, privacy, and mistracking. Intrusiveness is also a concern, as suites of multimodal sensors may be impractical in some settings, or they may adversely affect the natural behavior of visitors [32].

Transfer learning presents itself as a natural solution to this issue, as the various modalities in a multimodal modeling framework share a common predictive task. In particular, recent years have seen an increased emphasis on domain adaptation, a type of transfer learning that investigates the predictive capacity of models that are pre-trained on one domain (*source* domain) and are subsequently reweighted to perform similarly on another domain with a different distribution (*target* domain) across a single common task [39]. A

Nathan Henderson, Wookhee Min, Andrew Emerson, Jonathan Rowe, Seung Lee, James Minogue and James Lester "Early Prediction of Museum Visitor Engagement with Multimodal Adversarial Domain Adaptation". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 93-104. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

primary objective of domain adaptation is to obtain a domain-invariant representation of the salient features extracted from the two distinct data sources, where the shared feature space allows for improved predictive performance on data points from the target domain while still maintaining strong performance on data from the source domain. Examples of recent domain adaptation research include adapting across images extracted from different domains [34, 42] or across modalities captured from different data channels such as RGB-to-depth image translation [33, 42].

In this work, we investigate the use of domain adaptation as a method of translating unimodal, interaction-based data to a domain-invariant representation that can be used with predictive models previously trained on multimodal data. We demonstrate the effectiveness of a multimodal domain adaptation framework for making early predictions of visitor dwell times at an interactive museum exhibit. Our multimodal analytics framework is designed to operate in museum settings where sensor-based data capture may be restricted or otherwise impractical. We adopt an adversarial approach to generating domain-invariant representations of multimodal data (exhibit interactions and facial expression serving as the *source* domain) and unimodal data (exhibit interactions serving as the *target* domain) that are encoded using stacked denoising autoencoders. Empirical results of evaluations of the framework suggests that the use of adversarial discriminative domain adaptation allows for a unimodal target encoder to be trained to share a latent feature space with a multimodal source encoder [42]. The framework achieves higher performance than an interaction-only baseline model in terms of early prediction and visitor-level prediction of dwell time, a proxy indicator of visitor behavioral engagement with an exhibit. Dwell time has been frequently used to quantify visitor engagement in museum settings [5, 23]. The framework offers the potential to accurately predict visitor dwell time in museums, while also allowing for operation with reduced availability of physical sensor data, or even when no physical sensor data is available.

2. RELATED WORK

Visitor engagement is a critical aspect of learning in informal learning environments, such as science centers and museums [21]. Engagement shapes how visitors proceed throughout a museum, and interact with various exhibits [16]. There has been substantial work on modeling engagement in formal learning environments such as classrooms [19] and laboratories [8], and this focus has expanded in recent years to informal learning environments. This includes research efforts focused on analyzing engagement in groups of visitors around interactive tabletop exhibits [5], investigating the effectiveness of interventions for enhancing group engagement at different diorama exhibits [23], and predicting visitor dwell time [16]. However, devising computational models of museum visitor engagement remains a relatively unexplored area and presents distinctive challenges due to the free-choice nature of visitor learning in museums, creating a need for data-rich engagement modeling techniques.

Multimodal engagement modeling has shown significant promise as an engagement modeling approach due to its capacity to provide a data-rich multi-dimensional perspective on learner behavior [2]. In many cases, multimodal models lead to improved performance compared to models that utilize a single modality [19, 22, 32, 49]. Multimodal models have often utilized several diverse data channels when deployed in formal learning environments, including facial expression, posture, eye gaze, dialogue, and interaction trace data [40]. Facial expression data is commonly used in multimodal learner models of student affect [7] and performance [44]. Posture data has also been used for affect detection [22] as well as predicting learners' levels of engagement with Massive Online Open Courses (MOOCs) [9]. Eye gaze data has been combined with facial expression and head pose data to train models for continuous emotion prediction [48], while dialogue data has been utilized to predict dropout in online K-12 courses [26]. Finally, interaction trace logs and keystroke data have been used in conjunction with facial expression data to detect confusion in students engaging with an introductory computer science education learning environment to provide adaptive feedback and support dynamic adjustment of exercise difficulty levels [6]. While recent work has investigated multimodal approaches to modeling visitor engagement in museums [16], multimodal approaches to museum visitor modeling poses significant challenges, as these frameworks often necessitate physical, sensor-based data capture. This introduces various ethical and logistical concerns and may be impractical or prohibitive in certain informal learning environments.

Computational methods such as transfer learning, and particularly domain adaptation, provide a way to harness the predictive capacities of multimodal learning analytics while allowing visitor modeling frameworks to operationalize a reduced number of more intrusive modalities. Domain adaptation and transfer learning have shown significant potential in a variety of implementations, and have been utilized within educational contexts for tasks such as confusion detection in online forums for different online courses [50] and automated essay scoring across different prompts [35]. Additionally, domain adaptation has been investigated within multimodal contexts such as RGB and depth images [42], as well as video and audio modalities [36]. To our knowledge, adversarial domain adaptation has not been applied to unimodal and multimodal data to model learner engagement in museums.

Recent domain adaptation work has focused primarily on an unsupervised or semi-supervised variation of this problem, where deep learning models trained on a labeled source dataset are transferred to share latent representations alongside a target domain that may contain little or no previously labeled data. The issue of missing labels for the target domain data is addressed by obtaining a domain-invariant representation through minimizing the distance between the learned data representations between the two domains [17, 41, 42]. While prior efforts accomplish this task through statistical measures such as the Maximum Mean Discrepancy (MMD) [43] or the deep Correlation Alignment (CORAL) [39], other work has taken an adversarial approach, with the simultaneous goals of learning a data representation that is predictive of the source domain labels while also being indistinguishable to a domain discrimination model [27, 42]. One approach involves reversing the gradients of a domain discrimination model to maximize the model's loss and guide the learning to explore a domain-invariant representation [17]. Other approaches train a source encoder to reduce the source domain data to a latent representation and use a domain discriminator to adversarially train a target encoder to produce a latent representation of the target domain data that is indistinguishable to the discriminator [42]. The trained target encoder is subsequently used to process unlabeled data from the target domain to be classified by a model pre-trained on source data. Another approach is the Co-GAN approach, which involves two Generative Adversarial Networks (GANs) that generate source and target data, respectively [27]. The high layer parameters of the two GAN models are tied together, allowing the generators of the models to co-learn a shared latent representation while possibly sharing a common input noise vector.

Early prediction is an important component of visitor modeling because it can drive run-time adaptive support to enhance visitor interest and engagement with interactive exhibits. A critical objective in early prediction is to reach a certain accuracy threshold in a timely manner. Early prediction has been investigated in the context of formal learning environments, such as predicting middle-grade learner engagement with a game-based learning environment [47], evolving learning goals throughout students' interaction trajectories [31], and student success in novice programming tasks [29]. Early prediction has also been the subject of prior work on museum learning, such as automatic detection of visitors' social behavioral patterns [13, 24] and multimodal regression-based modeling of visitor engagement in science museums [16].

The primary contributions of this work are as follows: (1) we demonstrate improved predictive performance of multimodal models of museum visitor dwell time using facial expression and interaction data compared to interaction-only baselines, (2) we evaluate the effectiveness of adversarial discriminative domain adaptation as a means of enabling the use of previously-trained multimodal models with unimodal data, and (3) we investigate the performance of each visitor engagement model using convergence-based early prediction metrics and standard predictive performance measures. Domain adaptation has been relatively underexplored with educational data, and this is especially true of data from informal learning environments such as museums. Furthermore, there has been limited work investigating domain adaptation in the context of early prediction of learner engagement. Our work shows that domain adaptation is effective at enhancing prediction of visitor dwell time by harnessing the capacities of multimodal visitor modeling, which leads to higher predictive accuracy when compared to unimodal models.

3. FUTURE WORLDS EXHIBIT

To investigate multimodal predictive models of museum visitor engagement, we use data collected from visitor interactions with a game-based museum exhibit, FUTURE WORLDS, which is designed to introduce visitors to concepts about environmental sustainability (Figure 1). FUTURE WORLDS runs on a multi-touch display, enabling visitors to interact with the virtual environment through touch and gestures on the screen. Visitors are faced with the challenge of improving the conditions of the virtual environment's biosphere through a series of changes such as farming practices and energy sources within the game. FUTURE WORLDS and its integrated educational content are targeted towards learners ages 10-11.

Visitors can tap or swipe on the screen to perform certain actions such as reading about a particular aspect of the virtual environment and its impact on sustainability or modifying an in-game element and observing the broader consequences of this decision on the environment. Upon making a change to the virtual environment, the visitor is given immediate feedback regarding the positive or negative impact of the gameplay action. A visitor can "win" by making the correct decisions to certain in-game elements that maximize the environmental sustainability of the virtual environment. Afterwards, the visitor is presented with the option to restart the game or continue interacting with the virtual environment in its completed state. Additionally, a visitor is able to leave the FUTURE WORLDS exhibit having not completed the game beforehand. Prior work with FUTURE WORLDS found that visitors improved their understanding of environmental sustainability

Figure 1. Gameplay of the FUTURE WORLDS interactive exhibit, including (A) 3D virtual environment, (B) selecting an element to modify, (C) viewing information about the selected element, and (D) correctly solving the in-game problem.

concepts, while also demonstrating high levels of engagement throughout their interactions with the exhibit [37].

4. MULTIMODAL DATA COLLECTION

To track visitor engagement and behavior with FUTURE WORLDS, the exhibit was instrumented with several sensors to collect the real-time behavior of visitors' interactions with the exhibit, as shown in Figure 2. We first describe the visitor population for study participants and then introduce the two modalities used for the domain adaptation approach (facial expression, exhibit interaction trace logs), and the features extracted from each input data channel.

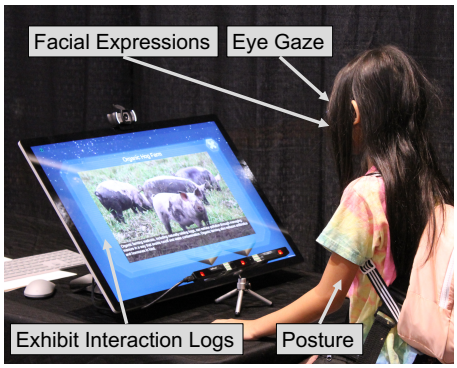


Figure 2. Visitor interacting with FUTURE WORLDS.

4.1 Study Participants and Procedure

We conducted a study of visitor interactions with the FUTURE WORLDS exhibit at the North Carolina Museum of Natural Sciences in Raleigh, North Carolina. The data were collected over a series of three sessions with different school groups of visitors aged 10-11 (M=10.4, SD=0.57). The school groups came from different socio-cultural backgrounds (e.g., race/ethnicity), and each school served student populations where 70% of the students are from low-income families. In total, 116 visitors interacted with FUTURE WORLDS. There were 47 female and 55 male participants, with 14 participants who did not provide data on their gender. The visitors were 32.4% Hispanic or Latino, 21.6% African American, 11.8% American Indian, 8% Asian, 7.5% mixed race, 3% Caucasian, and 15.7% preferred not to respond. Before interacting with the exhibit, visitors were asked to complete a series of surveys and questionnaires, including a demographic survey, sustainability content knowledge assessment, and the Fascination in Science scale [11]. Afterward, visitors interacted with the exhibit until they wanted to stop or after approximately 12 minutes had elapsed (M=5.8, SD=2.4, Min=1.8, Max=11.8). Visitor dwell times were captured by the game's internal logging functionalities. Once visitors finished their interaction with the exhibit, they were asked to complete a sustainability content knowledge assessment, engagement survey, and a short debrief interview. Several visitors were missing one or multiple data channels (e.g., facial mistracking), requiring the removal of their data from the final dataset for analysis. The final dataset that was used for the predictive models in this paper consisted of multimodal data from 79 visitors.

During the data collections, the visitors' body movement, eye gaze, facial expression, and interaction data from the exhibit were captured. For this study, we focus exclusively on the exhibit interaction data and the facial expression data. We selected the exhibit interaction data due to its unintrusive nature and its relative ease of data capture, as the trace data is captured in the background with the exhibit software and does not require any physical sensors or calibration. We selected facial expression data because of its predictive utility in previous work on unimodal and multimodal models of learner engagement [14, 15].

4.1.1 Facial Expression

Facial expression is an important indicator of learner emotion, and it has been widely used in previous studies on modeling learner engagement [46]. In this work, visitor facial expression was captured using video data from an externally mounted Logitech C920 USB webcam. In real time, the video data was processed by OpenFace, an open-source facial behavior analysis toolkit to detect facial landmarks, estimate head pose, recognize facial action units (AUs), and estimate eye gaze [3]. The OpenFace software automatically detects and analyzes 17 distinct AUs for each visitor's face captured within the camera's field of view.

4.1.2 Interaction Trace Logs

FUTURE WORLDS includes built-in logging functionalities to capture fine-grained logs of visitor interactions with the exhibit. The interaction trace logs consist of sequential records (at the millisecond level) of physical interactions with the multi-touch display (e.g., taps, swipes, and gestures), as well as specific in-game learning events (e.g., altering the virtual environment and accessing an embedded informational resources). The interaction trace logs are used to investigate how visitors interacted with the exhibit and progressed through the game.

4.2 Multimodal Features

Using both visitors' facial expression and exhibit interaction behavior, we distilled two sets of features to serve as predictors of visitor dwell time. Many of the extracted features for each modality were chosen based on their predictive performance in prior work on multimodal learning analytics [16].

4.2.1 Facial Expression

Using the processed AU data from OpenFace, we calculated the duration that each AU was exhibited throughout the visitor's interaction with FUTURE WORLDS. We first standardized each visitor's observed AU intensity values and then calculated the duration of each AU during time intervals where consecutive AU intensity values were at least one standard deviation greater than the mean of that particular visitor-specific AU feature. This filtering process ensured that only spikes relative to the specific visitor's AU values contributed towards the calculation of the total duration. To further filter the AU durations, we only recorded the duration if the AU was present for longer than 0.5 consecutive seconds. This avoided possible micro-expressions that could add noise to the overall data channel [38]. We performed this filtering process for all 17 AUs tracked by OpenFace. In addition, we generated the standard deviation and maximum AU values across the visitor's interactions up to the current timestamp. In total, we extracted and distilled 51 facial expression-related features.

4.2.2 Exhibit Trace Logs

We distilled eight features from the exhibit interaction data: (1) the total number of times a visitor tapped the FUTURE WORLDS multi-touch display, (2) the total number of times a visitor tapped informational tiles about environmental sustainability concepts, (3) the total duration of time an informational tile was open, (4) the total duration spent with labeled sustainability images displayed onscreen, (5) the total duration of time that a visitor spent directly interacting with the 3D simulated environment in FUTURE WORLDS, (6) the total number of times a visitor swiped the interface to explore alternative options for modifying the simulated

environment, (7) the total number of times the simulated environment was modified, and (8) a binary feature that indicated whether a visitor had successfully solved the current environmental problem scenario in FUTURE WORLDS.

5. DOMAIN ADAPTATION

In this work, we present an unsupervised, adversarial discriminative domain adaptation approach that enables the use of multimodal visitor engagement models in settings where only unimodal data streams are available. In unsupervised domain adaptation, two datasets are extracted from two separate domains: (1) a source domain (s), from which data samples X_s and associated labels Y_s are drawn, and (2) a target domain (t), which contains unlabeled data samples X_t. It is also assumed that there exists a classifier C_s that has been previously trained on the source data X_s and source labels Y_s by learning a latent mapping M_s. The primary objective of the unsupervised domain adaptation approach is to learn a latent mapping M_t so that $M_t(X_t)$ can be correctly classified by C_s despite the absence of any associated labels for X_t.

The purpose of adversarial training within the domain adaptation framework is to learn a domain-invariant data representation that minimizes the distance between $M_t(X_t)$ and $M_s(X_s)$. This is accomplished through a separate binary discriminator, D, that is trained to distinguish between latent representations of the source domain and the target domain. The discriminator is optimized according to a standard cross-entropy loss function (Equation 1):

$$\mathcal{L}_{DISC}(X_s, X_t, M_s, M_t) = -\mathbb{E}_{x_s \sim X_s}\left[\log D\left(M_s(X_s)\right)\right] - \mathbb{E}_{x_t \sim X_t}\left[\log\left(1 - D\left(M_t(X_t)\right)\right)\right] \quad (1)$$

Adversarial domain adaptation focuses on two primary objectives implemented within a minmax framework: the discriminator attempts to accurately classify a latent data representation as either from the source domain or the target domain, while a target encoder attempts to learn a mapping $M_t(X_t)$ that deceives the discriminator, thus finding a latent representation that is domain-invariant but retains enough salient characteristics to provide predictive value to a source classifier C_s. To implement an adversarial loss function within the framework, a common practice is to simply invert the loss term when training the target encoder. This essentially reverses the gradients for the target encoder but can consequently lead to premature convergence and vanishing gradients [17]. A more stable training method is to invert the labels used to train the target encoder. This creates two distinct convergence objectives for the two elements of the adversarial framework [42]. The discriminator loss term remains the same as stated in Equation 1 above, while the loss term for the target encoder becomes:

$$\mathcal{L}_{TAR}(X_s, X_t, D) = -\mathbb{E}_{x_t \sim X_t}\left[\log D\left(M_t(X_t)\right)\right] \quad (2)$$

This process is analogous to the process utilized by generative adversarial networks (GANs) [18]. A GAN attempts to emulate a fixed data distribution by adversarially training a discriminator to distinguish between "fake" data, which was produced by a generator that aims to generate data that is synthetic but realistic looking using a random noise vector, and "real" data that is extracted from the prior fixed data distribution. While GANs have been utilized in domain adaptation tasks [27], they are typically effective when the source and target domains are relatively similar. GANs have shown convergence issues in scenarios involving a high degree of domain shift [42]. As our work involves a domain shift from a multimodal source domain to a unimodal target domain, we opt to utilize a non-generative approach for this work and focus exclusively on discriminative adversarial methods. It is assumed that a pre-existing distribution of multimodal data (i.e., interaction trace logs + facial expression) is available to train the source encoder and the source classifier, while the target distribution consists of unlabeled unimodal data (i.e., interaction trace logs). This is intended to simulate scenarios where visitor engagement models have been previously trained on multimodal data but are deployed in situations where only interaction trace log data is available to generate new predictions of visitor engagement.

While much prior work in adversarial domain adaptation involves source and target domains of similar or identical dimensionality (e.g., image-to-image translation), the multimodal aspect of this work presents a distinct challenge, as the multimodal data in the source domain inherently contains more features than the unimodal target domain. To enable the pre-trained multimodal classifier to handle unimodal data as input, stacked denoising autoencoders [45] are used to reduce the multimodal and unimodal feature vectors to the same dimensionality. An autoencoder is an unsupervised method of using feedforward neural networks to reduce an input vector X to a latent data representation using an encoder that contains a mapping function M. The autoencoder then attempts to use a decoder that uses mapping function N to reconstruct $M(X)$ to its original input. The encoder and decoder components of the autoencoder are both optimized by minimizing the reconstruction loss between X and $N(M(X))$. A stacked autoencoder is a variation in which each component contains multiple hidden layers of autoencoders. A denoising autoencoder builds on the same concept but corrupts the input vectors using random noise injection, which allows effective model regularization [45]. In this work, we use a corruption level of 0.25 on each feature in each input vector, where a value is set to 0 when the input feature is corrupted. After input vector X undergoes random noise injection to produce X', the denoising autoencoder attempts to reconstruct X from $N(M(X'))$. This allows the autoencoder to become more robust against random noise within the input features while also preventing the autoencoder from overfitting or simply learning the identity function. Following the optimization of the autoencoder, the decoder component is discarded while the encoder component is retained for dimensionality reduction within our data processing pipeline. A denoising autoencoder is shown in Figure 3.

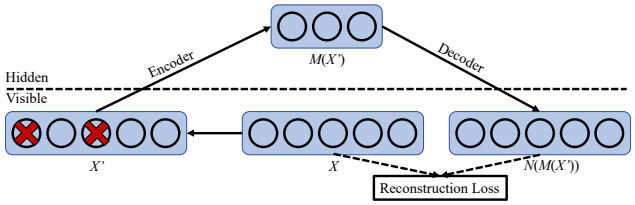


Figure 3. A denoising autoencoder.

Our adversarial domain adaptation process is shown in Figure 4. Figure 4A illustrates the initial training of the classifier and the source encoder. The features from the facial expression and interaction modalities are concatenated together and then used to train a stacked denoising autoencoder. Following this process, the trained source encoder is then used to reduce the multimodal input data to a latent representation that is then used to train a classifier. The classifier receives the latent data as input and is trained to predict the target variable, visitor dwell time. To enable the adversarial training of the target encoder and discriminator (Figure 4B), the weights of the pre-trained source encoder are fixed, and the target encoder weights are initialized using a pre-trained autoencoder optimized on the unlabeled, interaction-only data. An

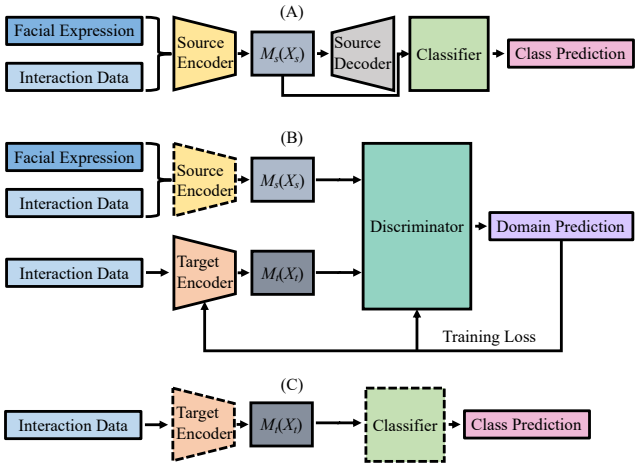


Figure 4. Domain adaptation process, including (A) the classifier and source encoder training, (B) adversarial training of the target encoder and discriminator, and (C) evaluation of the adapted target encoder on the classifier. Dashed lines indicate fixed model weights.

alternative approach is to initialize the target encoder weights from the source encoder. However, this can only be accomplished if the feature vectors extracted from the source domain are the same dimensions as the target domain. In our work, the multimodal feature vectors from the source domain have a higher dimensionality than the unimodal feature vectors from the target domain, since we remove the facial expression modality from the training data for the target encoder. The source and target encoders are used to produce latent representations of the multimodal and unimodal features, respectively. These representations are assigned labels of either 1 if the sample originated from the source domain, and 0 if the sample originated from the target domain. The data/label pairs are then used to train a feedforward network serving as the discriminator model. The discriminator is trained to distinguish between latent data from the source domain and from the target domain, while the target encoder is simultaneously trained to produce latent data from the target domain that consistently deceives the discriminator. To evaluate the target encoder (Figure 4C), unimodal data is passed through the encoder, and the resulting encoded data is then forward propagated through the trained classifier shown in Figure 4A. This procedure provides a way to evaluate the predictive performance of a multimodal classifier on unimodal data. It is important to note that some amount of multimodal data must be present prior to deploying our adversarial approach in order to train the multimodal classifier as well as the multimodal autoencoder.

6. METHODOLOGY

In multimodal models of learner engagement, some modalities that are highly predictive of engagement can also be impractical or undesirable in certain educational settings, such as sensors that require a cumbersome calibration process or expensive specialized equipment. Modalities that involve the capture of video data can raise concerns about privacy. However, eliminating physical sensors and exclusive reliance on sensor-free modalities may result in decreased performance on some tasks and settings. We propose a solution to this issue that (1) allows the predictive capacities of multimodal models to be retained, and (2) allows for the reduction in use of physical sensors. This work operates under the assumption that multimodal data is available in at least some capacity to facilitate the training of multimodal models prior to adversarial domain adaptation. As a result, the ideal setting for the proposed framework is after an initial multimodal data collection has taken place, enabling pre-trained multimodal models to be deployed. Below we describe the methods used to preprocess the multimodal and unimodal data, the feature selection process utilized to select the data used in the prediction and adversarial tasks, and the approach to training and validation of the visitor engagement models. Finally, we present the early prediction convergence metrics used to evaluate the final classification models and the domain encoders.

6.1 Data Preprocessing

6.1.1 Temporal Feature Engineering

To facilitate early prediction of visitor engagement, sequential representations were produced from the features engineered from the two modalities as described in Section 4.2. To accomplish this, feature vectors were engineered for every subsequent 10-second interval in a single visitor's interaction session with the exhibit. For each feature, the average or sum of all values from t=0 to t=10n seconds was calculated, where n is the number of 10-second intervals that have elapsed for that feature vector. For example, if a visitor engaged with the exhibit for one minute, then n=6, and the feature vectors are generated across time intervals of 10, 20, 30, 40, 50, and 60 seconds from the beginning of their session. This allows each feature vector to be a representation of a visitor's behavior over their entire interaction with an exhibit up to that point. Additionally, this approach solves the issue of the temporal alignment of the separate data channels caused by differing sampling rates of the facial expression modality and the interaction-based modality. As a result, the early prediction models are able to make predictions at a consistent frequency across every visitor's exhibit interaction trajectory (i.e., every 10 second). To ensure that the additive nature of the features does not contribute to artificially inflated model performance, each feature is scaled by the elapsed time up to the current timestamp. After this process is complete, 2,279 data samples were generated for 79 visitors.

6.1.2 Visitor Dwell Time

The beginning of a visitor's dwell time takes place after a calibration process with the eye gaze sensor is completed, and prior to when they are presented with an on-screen information dialogue box explaining the problem to be solved. The visitor's session can end one of three ways: (1) the visitor opts to end their session prior to completing the problem-solving task in FUTURE WORLDS, (2) the visitor solves the problem and chooses to end their session, or (3) the visitor solves the problem, opts to continue interacting with the virtual environment, and later chooses to end their session. Each visitor's dwell time was captured in total seconds (M=268.83, SD=137.48, Min=77.11, Max=657.48) and was recorded by the FUTURE WORLDS exhibit's built-in logging functionalities. For the purpose of this work, the dwell time prediction task was converted to a classification problem by splitting dwell time into three tertile groups and assigning approximately one-third of the visitors to each group. We use this classification approach instead of regression analysis due to the relatively low number of visitors in the dataset and to accommodate the use of early prediction convergence metrics. The visitors in the dataset were assigned to one of three possible groups according to their dwell time d: low (d <= 193.54, N=26), low (193.54 < d <= 318.82, N=27), and high (d > 318.82, N=26). We take this approach as a way to prevent a significant class imbalance while still retaining a higher level of granularity than a median split. The distribution of visitor dwell times, including the ternary groups, is shown in Figure 5.

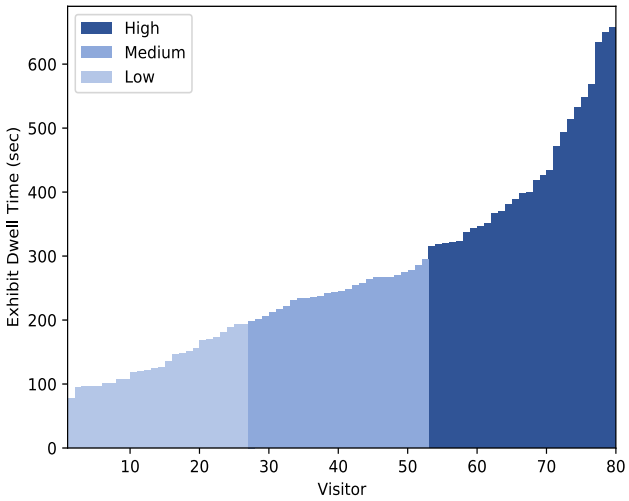


Figure 5. Distribution of visitor dwell times and ternary groups.

6.2 Feature Selection

Because of the large number of features in the multimodal data (51 facial expression features and 8 exhibit interaction features), we implemented forward feature selection to eliminate features with little or no predictive value and to reduce potential noise. Forward feature selection iterates through a list of features in a greedy manner, training a model on a single feature and continuing to add features if their inclusion increases the performance of the model on the target variable. This process continues until a predetermined number of features are selected or until all available features have been evaluated. This process has a few shortcomings. Due to the greedy nature of the algorithm, the features that are evaluated first have a higher chance of being selected. For example, the first feature that is evaluated is always retained, regardless of its true contribution to the predictive performance of the model. One approach to mitigating this issue is to perform forward feature selection for every possible combination of features, but this is often prohibitive as the number of combinations increases exponentially as the number of features increases, which imposes significant computational requirements. To mitigate the issue of bias in greedy feature selection while avoiding an exhaustive search across all feature combinations, we perform forward feature selection across a randomized ordering of all available features. We used a support vector machine (SVM) as the predictive model for each feature combination due to its effectiveness in high-dimensional spaces and relatively small computational overhead. This process was repeated for 100 separate iterations and randomizations to ensure that each feature had an equal probability of being placed at a specific point within each feature ordering. Following this process, the features were sorted according to the frequency that each feature was selected across all 100 iterations. To compensate for the difference in the number of features for each data channel, we performed forward feature selection on the facial expression modality and selected the ten most frequently selected features.

It should be noted that because we selected the ten most frequent features from the facial expression modality, and the interaction-based modality contained only 8 total features, each feature from the latter modality was included in the data modeling process. (We perform forward feature selection on the interaction-based features for analysis purposes only.) Because certain features such as AU durations and tile durations increase monotonically throughout a visitor's exhibit interaction trajectory and can lead to indirect data leakage with regard to the target variable (dwell time at the exhibit), the features were scaled by the total elapsed time up to the current timestamp, so these features were converted to proportional representations of the elapsed time at each time interval.

Table 1. Most frequent features from forward feature selection (interaction)

Feature	Frequency
Proportional Tile Duration	0.637
Proportional Open Tile Count	0.561
Proportional Info Duration	0.557
Proportional Info Taps	0.554
Proportional Taps	0.511
Proportional Swipe Tiles Count	0.416
Proportional Modify Tile Count	0.341
Beat Game	0.272

Table 2. Most frequent features from forward feature selection (facial expression)

Feature	Frequency
AU05 Max	0.317
AU10 Max	0.276
Proportional AU10 Duration	0.257
AU02 Max	0.237
Proportional AU01 Duration	0.227
AU26 Std	0.218
AU25 Max	0.214
Proportional AU17 Duration	0.208
Proportional AU45 Duration	0.206
Proportional AU26 Duration	0.196

This feature selection process took place within each cross-validation fold, and as a result, each fold produced a different combination of selected features. We calculated the frequency of the features across all cross-validation folds and present these in Table 1 and Table 2.

Based on the results in Table 1, features related to general interactions (proportional number of times any tile was opened, proportion of time any tile was open) were the most predictive interaction-based features. The features related to opening and viewing embedded graphical and textual science materials were also frequently selected features. The features representing the frequency a visitor modified the in-game virtual environment were less frequently selected as predictive features, as was the binary indicator of whether the visitor correctly solved the problem at that particular timestamp.

The most predictive features from the facial expression modality were primarily maximum values and proportional durations of certain AUs. AU05 (upper lid raiser) and AU10 (upper lip raiser) were the most predictive facial action units, followed by AU02 (outer brow raiser) and AU01 (inner brow raiser). AU25 (lips part) and AU26 (jaw drop) were moderately predictive, followed by AU17 (chin raise) and AU45 (blinking). Multiple representations

of AU10 and AU26 were frequently selected during the feature selection process as well. It is notable that the overall frequency of the facial expression features is significantly lower than many interaction-based features. This is likely a result of the large number of facial expression features compared to the interaction-based features.

6.3 Model Evaluation

The models were evaluated using 10-fold cross-validation, with the splits for each fold occurring at the visitor level to ensure that a visitor's data was contained only within a single training, validation, or test set. The dataset was standardized within each cross-validation fold by dividing each feature by subtracting the feature's mean and dividing by the feature's standard deviation, as determined by the training data. This rescales the data to have a standard deviation of 1 (unit variance) while centering the mean to be 0. Following this process, class imbalances within the training data were resolved using Synthetic Minority Oversampling Technique (SMOTE) [10]. SMOTE is a common upsampling approach that resolves class imbalances through a randomized K-nearest neighbor approach, which brings the class balance to a uniform distribution while avoiding duplication of any data points. The upsampled, standardized training data is then used for forward feature selection as described in Section 6.2.

After feature selection, a classifier model was trained on the multimodal data and the visitor dwell time labels in each cross-validation fold to provide a comparison point for the domain-adapted models. The tertile labels for the target variable were encoded as one-hot vectors for each model output. We evaluated five different models: SVM, logistic regression, naïve Bayes, random forest, and a feedforward neural network. We performed hyperparameter tuning using a 3-fold nested cross-validation within the training set for each outer cross-validation fold. The hyperparameters that were varied for each model included the regularization parameter and kernel (SVM), regularization parameter (logistic regression), number of estimators (random forest), and number of layers and nodes (feedforward neural network). Additionally, the architecture of the autoencoder used to train the source encoder was evaluated during the hyperparameter tuning phase. The autoencoder was a feedforward neural network, and the hyperparameter values that were evaluated were the number of layers and nodes in the hidden layers within the encoder and decoder, as well as the number of latent dimensions. The feedforward neural network achieved the optimal performance as the classifier for visitor dwell time, using two hidden layers with 64 nodes each. The source encoder contained three hidden layers with 64, 32, and 16 nodes, respectively, with a latent output of 10. Additionally, all feedforward neural network models used a learning rate of 0.001, a dropout rate of 0.5 in the last hidden layer, and sigmoid activation functions. The loss function used for each model was categorical cross-entropy. Early stopping was implemented for each model using the validation data during the nested cross-validation to protect against overfitting. As a baseline, we follow the same process previously described, except using only the interaction modality. We evaluate both a unimodal and multimodal baseline in order to demonstrate the improved performance of the multimodal model of visitor dwell time as compared to the unimodal model, and to show the improved performance using the domain adaptation framework in situations where only unimodal data is available.

After the optimal classifier and source encoder for adversarial domain adaptation were trained for each cross-validation fold, the models' weights were fixed to evaluate the classifier performance on interaction-only data and to encode the multimodal data within the adversarial framework, respectively. The adversarial framework used a target encoder that is a feedforward neural network whose architecture and weights were pre-determined using the interaction-only baseline model. Although the source encoder and target encoder weights were not tied together as is common in other adversarial domain adaptation work [27], there was an imposed restriction that the latent dimensions be the same for both domains due to the fixed input size of the discriminator. The discriminator in the adversarial framework was a feedforward neural network with two layers of 64 nodes each. The learning rate of both the discriminator and the target encoder was 0.001, with a dropout rate of 0.05 in the last hidden layer and hyperbolic tangent activation functions. The loss functions for the discriminator and target encoder were based on binary cross-entropy as shown in Equations 1 and 2, respectively. The adversarial domain adaptation took place within each cross-validation fold to prevent data leakage from the test set.

To evaluate the predictive performance of the domain-adapted representations of the target data, the trained target encoder was used to encode the interaction-only data from the held-out test set within each cross-validation fold, and the encoded data was passed to the classifier model trained with the source data. The predictive performance of the classifier on this data was used to confirm that the use of multimodal data to train the classifier induces higher performance than if the facial expression data was removed from the dataset entirely. As an additional baseline, the target encoder trained on the interaction-only modality was used to pass the encoded data directly to the multimodal classifier without the domain adaptation procedure, following the source-only baseline approach of Tzeng et al. [42]. This illustrates that any improvement due to our method can be attributed to the adjusted weights through the adversarial adaptation process instead of just compressing the latent representation of the target domain data to the source domain's dimensionality. This specific baseline is called *target-only*.

6.4 Early Prediction

To quantify the models' ability to accurately predict a visitor's dwell time early and consistently, we utilize two metrics: *standardized convergence point* [30] and *convergence rate* [4]. The *standardized convergence point* calculates an average point of model convergence to the correct labels, while a particular visitor's sequence not converged to a correct prediction is penalized. This metric extends the conventional *convergence point* metric to account for sequences that are ultimately predicted incorrectly and fail to converge by instituting a penalty term [4]. In this instance, standardized convergence point is greater than one. In cases of convergence, a sequence's standardized convergence point falls within the range [0, 1]. Equation 3 displays the formula used to calculate the standardized convergence point across all sequences, where m is the number of sequences, and n_i is the number of data points in the i^{th} visitor's sequence. The value of k_i is the number of data points after which the model makes consistently accurate predictions, otherwise k_i equals n_i+p_i, where p_i is the penalty term for the i^{th} sequence [30]. (p_i is set to 1 for all sequences in this work following the original work.) A lower standardized convergence point indicates that the model's predictive accuracy tends to converge earlier in a visitor's interaction with the exhibit, indicating better early prediction performance.

$$Standardized\ convergence\ point = \sum_{i=1}^{m} \frac{\left(\frac{k_i}{n_i}\right)}{m} \qquad (3)$$

The second metric that we use to quantify a model's early prediction performance is the *convergence rate*. Convergence rate is the percentage of observed sequences in which the final prediction is accurate. Any sequence that contains an accurate dwell time prediction at the last data point is considered to have converged. Therefore, a higher convergence rate is indicative of better performance.

7. RESULTS AND DISCUSSION

The results for the unimodal and multimodal models as well as the unimodal latent representations (i.e., target-only encoding) and domain-adapted representations are shown in terms of early prediction and visitor-level predictive performance in Table 3. To measure visitor-level performance, a single point estimate of the predictive performance for each individual visitor is obtained by averaging across the predictions for all data points. The results for Table 3 are shown in terms of standardized convergence point (SCP) and convergence rate (CR) for early prediction, and area under curve (AUC), Cohen's Kappa, accuracy, and F1 score for visitor-level performance. Although AUC is commonly used for binary classification problems, we use this metric for a multi-class approach using a "one vs. rest" method which treats the correct class as the "positive" group and combines all other classes as a single "negative" group. The total AUC for a single model is calculated by using the unweighted mean of the AUC values across all three groups.

Based on the results in Table 3, the adversarial domain adaptation allows the multimodal classifier to outperform all baselines in terms of early prediction and across all sequences for each visitor. As expected, the complete multimodal model achieved the highest performance, achieving an AUC value of 0.660, while also outperforming the other models in all other evaluation metrics. The model achieved a standardized convergence point of 64.58%, indicating that the model achieved and maintained its optimal predictive performance approximately 64% into a visitor's total dwell time at the exhibit, while converging to the correct predictions more often than other baseline approaches. The interaction-only modality produced noticeably lower performance, achieving a convergence point of 75.95%, while also reaching a 0.574 AUC across all sequences. The adversarial domain adaptation allowed the classifier to achieve higher performance on the interaction-only data, with an early prediction performance of 67.42% and a visitor-level AUC of 0.585, similar to the full multimodal model while also outperforming the interaction-only baseline across all evaluation metrics.

The classifier's performance on the latent unimodal data (without domain adaptation) was notably poor, achieving an AUC that was slightly worse than random chance (0.500). This result is not surprising, as we are evaluating the model's performance using latent representations from a domain that has not been used to train the model beforehand. Although similar baseline approaches can achieve moderate performance in instances where the source and target domains are relatively similar, other work that investigates cross-modality adaptation or adaptation across dissimilar domains achieves much lower performance for this specific baseline [42].

While the adversarial domain adaptation proved more effective than the interaction-only and latent unimodal data baselines, the performance of our framework did not achieve the same performance as a framework that contained the full multimodal data. This could be attributed to the significant difference between the interaction and facial expression domains. The majority of the interaction-based modality is comprised of discrete, monotonically increasing features, which inherently are not as data-rich as the features from the facial expression modality. Because there are multiple features for each AU, this modality provides multifaceted perspectives on multiple AUs, leading to a relatively high number of continuous features. Adapting between two data channels with such a discrepancy in dimensionality may be a contributing factor to the framework's performance. Second, the relatively small number of visitors in the dataset may also be a contributing factor, as the performance of the models could be at risk for overfitting the classifier, source encoder, or target encoder. Contributing to this potential issue is the loss induced in the domain adaptation process. The size of the dataset may prevent the adversarial framework from reaching optimal convergence. Third, because there is no restriction regarding how long the visitors could remain at the exhibit, the target variable has a relatively wide range of values, approximately from one minute to more than ten minutes. Although this issue is addressed through the use of a tertile split, additional data could provide further evidence of behavioral patterns that are able to induce higher performance with more granular target variables.

Because timestamped interaction trace logs are the basis of one of the modalities used in this work, the design of the museum exhibit may play a role in the performance of the visitor models in terms of early prediction. During the early stages of FUTURE WORLDS, visitors are prompted to read an information dialog box explaining the premise of the game and a summary of the problem to be solved in the virtual environment. Because this event occurs at the beginning of every visitor's interaction sequence, it is likely that more indicative behaviors that allow the classifier to differentiate between groups occur at later stages of learner interactions with the exhibit. This is a potential explanation behind the early prediction performance of each model, as the standardized convergence point occurs after 60% of the overall exhibit interactions across all models.

To further investigate the impact that domain adaptation has on the predictive performance of the multimodal classifier, confusion matrices based on the target-only encoder and the adversarially-trained encoder are shown in Figure 6 as is the confusion matrix for the interaction-only classifier. The purpose of this analysis is to determine if adversarial domain adaptation results in any changes relative to the classifier's sensitivity to certain dwell time groups.

Table 3. Visitor-level predictive performance (all sequences)

	Early Prediction			Visitor-Level Prediction			
Encoding	Classifier	SCP	CR	AUC	Kappa	Accuracy	F1 Score
Interaction-Only	Unimodal	75.95%	34.18%	0.574	0.085	0.392	0.355
Multimodal	Multimodal	**64.58%**	**48.10%**	**0.660**	**0.278**	**0.519**	**0.511**
Target-Only	Multimodal	73.79%	34.18%	0.499	0.015	0.342	0.338
Domain Adaptation	Multimodal	**67.42%**	**43.04%**	**0.585**	**0.203**	**0.468**	**0.468**

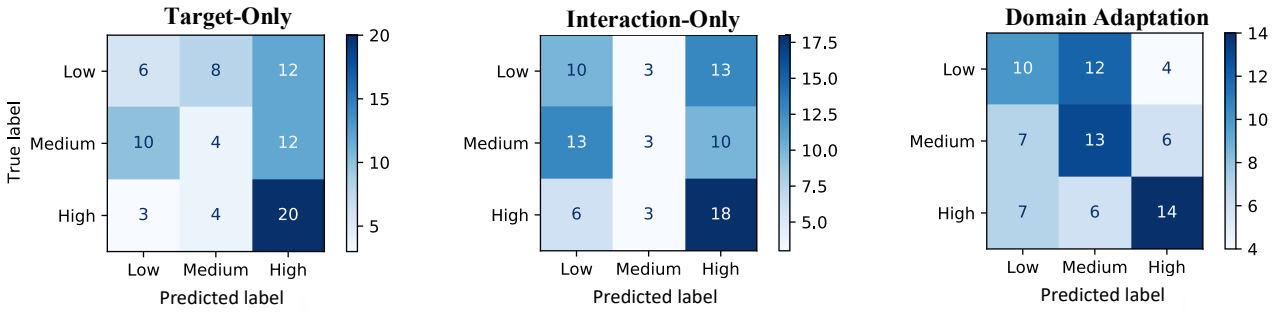


Figure 6. Confusion matrices for classifiers using target-only, interaction-only, and domain adaptation-based representations.

Based on the confusion matrix for the target-only classifier (i.e., the multimodal classifier evaluated on the interaction data without domain adaptation), the classifier appears to primarily predict high dwell times for a majority of visitors. The model also appears to frequently predict visitors with medium dwell time as having low dwell time. As this particular model performed similarly to a random chance classifier, it is likely that the interaction-only data representation was not easily identifiable to the classifier, leading it to primarily predict a single class and not classify the lower two groups accurately. The classifier that was trained and evaluated on interaction-only encodings performed slightly better and appears to become more accurate in cases of lower dwell time in visitors. However, it is notable that the model still does not appear to accurately predict instances of medium dwell time. This indicates that the interaction-based modality contains salient features indicative of noticeably low or high engagement but interactions from visitors with medium dwell time are not easily distinguishable to the model. Low dwell time may be characterized by a relatively low number of taps or interactions in the virtual environment, while high dwell time may be indicated by greater or more frequent tapping or interactions with the virtual environment. Additionally, visitors that have a higher dwell time are more likely to beat the game or read a higher number of information dialogs. However, this information may not be predictive enough with the ternary split, causing the interaction model to overfit to the two extremes.

The multimodal classifier that processes the modality-invariant data representations performs noticeably better for visitors with medium dwell time and continues to maintain fairly accurate performance on visitors with high dwell time. This may indicate that facial expression captures physical cues that allow the model to more easily distinguish between the medium group and the other groups, and the domain adaptation allows these features to be integrated into the interaction-only representations. By implementing this approach across the two modalities, it appears that the multimodal model retains its robustness to visitors with a medium dwell time in particular, while being able to achieve this performance using only features from the interaction data. This is significant because it appears that the interaction-only model does not appear to induce high performance on the medium dwell time visitors, so it remains important to utilize the multimodal data representations obtained through domain adaptation as pre-training for accurately predicting the visitor dwell time.

8. CONCLUSION

Modeling visitor engagement is an important task in museum-based learning. However, visitor engagement modeling presents significant challenges, as visitors' patterns of engagement with museum exhibits can vary widely. Multimodal frameworks show promise for the prediction of visitor engagement in museums because they capture information about visitor behavior that cannot otherwise be captured through interaction trace logs or similar unimodal data channels. Although multimodal sensor systems give rise to concerns about privacy, feasibility, and intrusiveness, the complete removal of sensor data from visitor engagement models may result in diminished predictive performance. To address this issue, we have introduced an adversarial domain adaptation approach to generating modality-invariant representations of interaction data and facial expression data from visitor interactions with the FUTURE WORLDS museum exhibit. The domain adaptation approach enables multimodal models to be induced in a pre-training phase while being deployed and evaluated with modality-invariant representations obtained using interaction-based data exclusively. We investigate the models' ability to predict visitor dwell time during the early stages of a visitor's interaction with the museum exhibit. Results indicate that the domain adaptation approach to modeling visitor engagement achieves higher performance than a visitor modeling approach using only a single modality. The domain adaptation approach also outperforms the unimodal baseline during early sequences of a visitor's interaction trajectory as well as across all sequences while demonstrating competitive performance compared to classifiers utilizing multimodal data.

There are several promising directions for future work. Alternative techniques for modeling visitor engagement should be evaluated, including sequential models like long short-term memory (LSTM) networks, to improve models' predictive accuracy and early prediction. Alternative approaches to the adversarial learning component of this framework include the use of generative models such as GANs or variational autoencoders. Attaining reliable training convergence continues to be a challenging problem within adversarial learning and investigating solutions to this issue may enhance the benefits of domain adaptation. The generalizability of the domain adaptation framework should be evaluated using larger and more diverse visitor populations on different exhibits and museum settings. Additionally, the domain adaptation framework should be evaluated using additional combinations of modalities (e.g., posture, gaze, speech), and extended to include three or more modalities simultaneously. Finally, this framework should be evaluated at run-time by integrating visitor engagement models into a museum exhibit to enable visitor-adaptive interventions to enrich visitor engagement and enhance museum-based learning experiences.

9. ACKNOWLEDGMENTS

The authors would like to thank the staff and visitors of the North Carolina Museum of Natural Sciences. This research was supported by the National Science Foundation under Grant DRL-1713545. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

10. REFERENCES

[1] Aslan, S., Alyuz, N., Tanriover, C., Mete, S., Okur, E., D'Mello, S. and Arslan Esme, A. 2019. Investigating the impact of a real-time, multimodal student engagement analytics technology in authentic classrooms. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[2] Baltrušaitis, T., Ahuja, C. and Morency, L. 2019. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 41, 2 (Feb. 2019), 423–443.

[3] Baltrusaitis, T., Zadeh, A., Lim, Y.C. and Morency, L. 2018. OpenFace 2.0: Facial behavior analysis toolkit. In *Proceedings of the 13th IEEE International Conference on Automatic Face Gesture Recognition.* 59–66.

[4] Blaylock, N. and Allen, J. 2003. Corpus-based, statistical goal recognition. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. 1303–1308.

[5] Block, F., Hammerman, J., Horn, M., Spiegel, A., Christiansen, J., Phillips, B., Diamond, J., Evans, E.M. and Shen, C. 2015. Fluid grouping: Quantifying group engagement around interactive tabletop exhibits in the wild. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 867–876.

[6] Bosch, N., D'Mello, S., Baker, R., Ocumpaugh, J. and Shute, V. 2015. Temporal generalizability of face-based affect detection in noisy classroom environments. In *Proceedings of the International Conference on Artificial Intelligence in Education*. 44–53.

[7] Bosch, N., D'Mello, S., Baker, R., Ocumpaugh, J., Shute, V., Ventura, M. and Zhao, W. 2016. Detecting student emotions in computer-enabled classrooms. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 4125–4129.

[8] Chan, M., Ochoa, X. and Clarke, D. 2020. *Multimodal Learning Analytics in a Laboratory Classroom*. Springer International Publishing.

[9] Chang, C., Zhang, C., Chen, L. and Liu, Y. 2018. An ensemble model using face and body tracking for engagement detection. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*. 616–622.

[10] Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W. 2002. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*. 16, 321–357.

[11] Chung, J., Cannady, M., Schunn, C., Dorph, R. and Bathgate, M. 2016. Measures Technical Brief: Fascination in Science.

[12] Diamond, J., Horn, M. and Uttal, D. 2016. *Practical Evaluation Guide: Tools for Museums and Other Informal Educational Settings*. Rowman & Littlefield.

[13] Dim, E. and Kuflik, T. 2014. Automatic detection of social behavior of museum visitor pairs. *ACM Transactions on Interactive Intelligent Systems*. 4, 4 (Nov. 2014), 17:1-17:30.

[14] D'Mello, S. and Kory, J. 2012. Consistent but modest: A meta-analysis on unimodal and multimodal affect detection accuracies from 30 studies. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*. 31–38.

[15] D'Mello, S. and Kory, J. 2015. A review and meta-analysis of multimodal affect detection systems. *ACM Computing Surveys*. 47, 3, 43:1-43:36.

[16] Emerson, A., Henderson, N., Rowe, J., Min, W., Lee, S., Minogue, J. and Lester, J. 2020. Early prediction of visitor engagement in science museums with multimodal learning analytics. In *Proceedings of the 2020 International Conference on Multimodal Interaction*. 107–116.

[17] Ganin, Y. and Lempitsky, V. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the International Conference on Machine Learning.* 1180–1189.

[18] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. 2014. Generative adversarial networks. *arXiv:1406.2661*. (Jun. 2014).

[19] Grafsgaard, J., Wiggins, J., Vail, A., Boyer, K., Wiebe, E. and Lester, J. 2014. The additive value of multimodal features for predicting engagement, frustration, and learning during tutoring. In *Proceedings of the 16th International Conference on Multimodal Interaction*. 42–49.

[20] Harrington, M. 2020. Connecting user experience to learning in an evaluation of an immersive, interactive, multimodal augmented reality virtual diorama in a natural history museum & the importance of the story. In *Proceedings of the 6th International Conference of the Immersive Learning Research Network.* 70–78.

[21] Hein, G. 2009. Learning science in informal environments: People, places, and pursuits. *Museums & Social Issues*. 4, 1 (2009), 113–124.

[22] Henderson, N., Rowe, J., Paquette, L., Baker, R. and Lester, J. 2020. Improving affect detection in game-based learning with multimodal data fusion. In *Proceedings of the International Conference on Artificial Intelligence in Education*. 228–239.

[23] Knutson, K., Lyon, M., Crowley, K. and Giarratani, L. 2016. Flexible interventions to increase family engagement at natural history museum dioramas. *Curator: The Museum Journal*. 59, 4 (2016), 339–352.

[24] Kuflik, T., Boger, Z. and Zancanaro, M. 2012. Analysis and prediction of museum visitors' behavioral pattern types. In *Ubiquitous Display Environments*. A. Krüger and T. Kuflik, Eds. Springer. 161–176.

[25] Lane, H., Noren, D., Auerbach, D., Birch, M. and Swartout, W. 2011. Intelligent tutoring goes to the museum in the big city: A pedagogical agent for informal science education. In *Proceedings of the International Conference on Artificial Intelligence in Education*. 155–162.

[26] Li, H., Ding, W., Yang, S. and Liu, Z. 2020. Identifying at-risk K-12 Students in multimodal online environments: A machine learning approach. In *Proceedings of the 13th International Conference on Educational Data Mining*. 137–147.

[27] Liu, M.-Y. and Tuzel, O. 2016. Coupled generative adversarial networks. *arXiv:1606.07536*. (Sep. 2016).

[28] Long, D., McKlin, T., Weisling, A., Martin, W., Guthrie, H. and Magerko, B. 2019. Trajectories of physical engagement and expression in a co-creative museum installation. In *Proceedings of the 2019 Conference on Creativity and Cognition*. 246–257.

[29] Mao, Y., Zhi, R., Khoshnevisan, F., Price, T., Barnes, T., and Chi, M. 2019. One minute is enough: Early prediction of student success and event-level difficulty during novice programming tasks. In *Proceedings of the 12th International Conference on Educational Data Mining.* 119-128.

[30] Min, W., Baikadi, A., Mott, B., Rowe, J., Liu, B., Ha, E.Y. and Lester, J. 2016. A generalized multidimensional evaluation framework for player goal recognition. In *Proceedings of the 12th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.* 197-203.

[31] Min, W., Mott, B., Rowe, J., Taylor, R., Wiebe, E., Boyer, K. and Lester, J. 2017. Multimodal goal recognition in open-world digital games. In *Proceedings of the AAAI Conference*

on Artificial Intelligence and Interactive Digital Entertainment. 13, 1 (Sep. 2017).

[32] Müller, P.M., Amin, S., Verma, P., Andriluka, M. and Bulling, A. 2015. Emotion recognition from embedded bodily expressions and speech during dyadic interactions. In *Proceedings of the 2015 International Conference on Affective Computing and Intelligent Interaction.* 663–669.

[33] Munro, J. and Damen, D. 2020. Multi-modal domain adaptation for fine-grained action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 122–132.

[34] Murez, Z., Kolouri, S., Kriegman, D., Ramamoorthi, R. and Kim, K. 2018. Image to image translation for domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 4500–4509.

[35] Phandi, P., Chai, K. and Ng, H. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 431–439.

[36] Qi, F., Yang, X. and Xu, C. 2018. A unified framework for multimodal domain adaptation. In *Proceedings of the 26th ACM International Conference on Multimedia*. 429–437.

[37] Rowe, J., Lobene, E., Mott, B. and Lester, J. 2017. Play in the museum: Design and development of a game-based learning exhibit for informal science education. *International Journal of Gaming and Computer-Mediated Simulations*. 9, 3 (2017), 96–113.

[38] Sawyer, R., Smith, A., Rowe, J., Azevedo, R. and Lester, J. 2017. Enhancing student models in game-based learning with facial expression recognition. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization.* 192–201.

[39] Sun, B. and Saenko, K. 2016. Deep CORAL: Correlation alignment for deep domain adaptation. In *Proceedings of the ICCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision.* 443–450.

[40] Tiam-Lee, T.J. and Sumi, K. 2018. Adaptive feedback based on student emotion in a system for programming practice. In *Proceedings of the International Conference on Intelligent Tutoring Systems*. 243–255.

[41] Tzeng, E., Hoffman, J., Darrell, T. and Saenko, K. 2015. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision.* 4068–4076.

[42] Tzeng, E., Hoffman, J., Saenko, K. and Darrell, T. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 7167–7176.

[43] Tzeng, E., Hoffman, J., Zhang, N., Saenko, K. and Darrell, T. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv:1412.3474*. (Dec. 2014).

[44] Vail, A., Grafsgaard, J., Boyer, K., Wiebe, E. and Lester, J. 2016. Predicting learning from student affective response to tutor questions. In *Proceedings of the International Conference on Intelligent Tutoring Systems*. 154–164.

[45] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. and Manzagol, P. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research.* 11, 12 (December 2010), 3371-3408.

[46] Whitehill, J., Serpell, Z., Lin, Y., Foster, A. and Movellan, J. 2014. The faces of engagement: Automatic recognition of student engagement from facial expressions. *IEEE Transactions on Affective Computing*. 5, 1 (Jan. 2014), 86–98.

[47] Wiggins, J., Kulkarni, M., Min, W., Mott, B., Boyer, K., Wiebe, E. and Lester, J. 2018. Affect-based early prediction of player mental demand and engagement for educational games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 243-249.

[48] Wu, S., Du, Z., Li, W., Huang, D. and Wang, Y. 2019. Continuous emotion recognition in videos by fusing facial expression, head pose and eye gaze. In *Proceedings of the International Conference on Multimodal Interaction.* 40–48.

[49] Yang, J., Wang, K., Peng, X. and Qiao, Y. 2018. Deep recurrent multi-instance learning with spatio-temporal features for engagement intensity prediction. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*. 594–598.

[50] Zeng, Z., Chaturvedi, S., Bhat, S. and Roth, D. 2019. DiAd: Domain adaptation for learning at scale. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*. 185–194.

Behavioral Testing of Deep Neural Network Knowledge Tracing Models

Minsam Kim
Riiid
Seoul, South Korea
minsam.kim@riiid.co

Yugeun Shim
Riiid
Seoul, South Korea
yugeun.shim@riiid.co

Seewoo Lee
Riiid
Seoul, South Korea
seewoo.lee@riiid.co

Hyunbin Loh
Riiid
Seoul, South Korea
hb.loh@riiid.co

Juneyoung Park
Riiid
Seoul, South Korea
juneyoung.park@riiid.co

ABSTRACT

Knowledge Tracing (KT) is a task to model students' knowledge based on their coursework interactions within an Intelligent Tutoring System (ITS). Recently, Deep Neural Networks (DNN) showed superb performance over classical methods on multiple dataset benchmarks. While most Deep Learning based Knowledge Tracing (DLKT) models are optimized for general objective metrics such as accuracy or AUC on benchmark data, proper deployment of the service requires additional qualities. Moreover, the black-box nature of DNN models makes them particularly difficult to diagnose or improve when unexpected behaviors are encountered. In this context, we adopt the idea of black-box testing / behavioral testing from Software Engineering and (1) define desirable KT model behaviors to (2) propose a KT model analysis framework to diagnose the model's behavioral quality. We test-run the framework using three state-of-the-art DLKT models on seven datasets based on the proposed framework. The result highlights the impact of dataset size and model architecture upon the model's behavioral quality. The assessment results from the proposed framework can be used as an auxiliary measure of the model performance by itself, but can also be utilized in model improvements via data-augmentation, architecture design, and loss formulation.

Keywords

Knowledge Tracing, Deep Learning, Behavioral Testing, Model Validation

1. INTRODUCTION

Assessment is a central task in Education, as it is involved in meta-cognition [17], tracing the skill trajectory, recommendation of contents [36], adjustment of tutoring strategy [14], and grading [3, 24, 33, 35]. With the advent of online educational platforms, there is an increasing demand in building assessment models using the interaction history data of users. One approach to track the skill of users is Knowledge Tracing (KT), which is the task to model students knowledge based on their coursework interactions within an Intelligent Tutoring System (ITS) [7].

To tackle the KT problem, the recent EdNet Challenge in Kaggle has gathered a total of 3,406 teams, 4,412 participants, to submit 64,678 models. Participants trained KT models on the EdNet KT dataset [6], and the models were evaluated by the Area Under the Receiver Operating Characteristic Curve (AUC). The AUC of the top 5 models were 0.820, 0.818, 0.818, 0.817, 0.817, which are very similar, and all models were based on the Transformer Neural Network structure [38]. While neural network structures are usually designed to appropriate human intuitions, most models lack interpretability compared to classical models. Therefore, when evaluation results for black-box models do not vary significantly, it becomes unclear how to choose the best model for deployment. Also, while small quantitative difference in the objective function or model AUC might not hurt the users' perception of the model reliability, few adversarial decisions of the black-box model can dissuade the user's faith. [34] also note that the performances of black-box models that are trained for general metrics such as classification accuracy or AUC(Area Under receiving operator characteristic Curve) can be overestimated.

As a result, Deep Learning based Knowledge Tracing (DLKT) models are not frequently implemented in the education community due to potential risks arising from the lack of model interpretability. In this study, we propose behavioral testing as an approach to alleviate this problem. The contribution of this work are summarized as follows:

- We propose a novel testing framework to validate DLKT models using a test on behaviors. The idea is to define consistent and convincing behaviors to be desired on DLKT models.
- As an example of applying the framework, we benchmark three state-of-the-art DLKT models from the

Minsam Kim, Yugeun Shim, Seewoo Lee, Hyunbin Loh and Juneyoung Park "Behavioral Testing of Deep Neural Network Knowledge Tracing Models". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 105-117. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

proposed validation framework. Positive results highlight the reliability of DLKT models and encourages the model's adoption, while negative results point out the limitations of DLKT models and show spaces for improvement.

- We introduce methods to utilize evaluations from the framework to design and improve DLKT models.

2. RELATED WORKS

2.1 Knowledge Tracing

Knowledge Tracing (KT) is the task to predict the expected correctness of an interaction of a student to a question by modeling the student's knowledge from past interactions [7]. In this study, we formulate the KT task as follows: the interaction sequence of a user is denoted as $X^u = \{x_1^u, x_2^u, \cdots, x_T^u\}$ where $u \in U$ is the user index. To simplify the notations, we omit the user index u unless specified. Each interaction $x_t = (q_{i_t}, c_t)$ at step t is defined by the pair of question $q_{i_t} \in Q$ and correctness $c_t \in \{0, 1\}$ where $Q = \{q_1, q_2, \cdots, q_n\}$ denotes the set of all questions and i_t denotes the question index of step t. A KT model predicts the correctness probability $P(c_t = 1 | x_1, x_2, \cdots, x_{t-1}, q_{i_t})$ of an unseen interaction X_t at step t, where $X_t = x_1, x_2, \cdots, x_t$ is the first t interactions of an interaction sequence X.

Notation	Description
$u \in U$	User index
X	Interaction sequence of a single user($= X^u$)
x_t	Interaction at time step t
$q_j \in Q$	Question
i_t	Question index at time step t
c_t	Correctness at time step t for question q_{i_t}
$c_t^{(q_j)}$	Correctness at time step t for question q_j
X_t	Interaction sequence of X up to time step t

Many KT models utilize domain specific tags such as skill components of questions [27, 8, 31, 43, 42], difficulty of questions [32, 10, 37], or knowledge graphs [4]. Item Response Theory (IRT) [32] models the correctness probability of a student responding to a question using custom designed models, and fits the model parameters using maximum likelihood. For instance, the 4-PL model predicts the correctness probability of a user with skill level θ solving item i by

$$p_i(\theta) = c_i + \frac{d_i - c_i}{1 + e^{-a_i(\theta - b_i)}},$$

where a_i, b_i, c_i, d_i are parameters that model discrimination, difficulty, pseudo-guessing, and slip of item i [23].

Another prominent approach is Bayesian Knowledge Tracing (BKT) [27, 42], which uses Markov process to model difficulty of the question items and learning capability of the students. Another well known approach is Deep Knowledge Tracing (DKT) [31], which is the first Deep Learning based KT model (DLKT). Since the introduction of DKT, many researchers have worked on different network structures to capture the complex aspects of the knowledge state. There are a variety of models based on different structures such as DKVMN [43], DKT+ [41], SKVMN [1], SAKT [30], GKT [28], EKT [21], KTM [39], DHKT [40], SAINT [5], AKT [13], and PEBG [22].

While there exist a variety of different KT models, [12] performed a major experiment on the accuracy of three groups of KT models (Markov process, Logistic Regression, Deep Learning) on nine real-world datasets. While deep learning models do show better AUC and RMSE on some datasets, other linear models including the authors' proposed BestLR approach yielded comparable or superior performances on most datasets, which also provided better model interpretability as well.

2.2 Behavioral Testing in Other Applications

To alleviate unexpected behaviors of black-box models, [2] introduces behavioral testing (also known as black-box testing) to test different capabilities of a system in the software engineering perspective. Many studies work on effectively designing test cases [18, 25, 26]. [29] gives a detailed review on the behavioral testing method applied in various software testing domains. In Natural Language Processing, [34] apply the behavioral testing framework to validate the behaviors of general NLP models. They introduce *CheckList*, which is a task agnostic methodology for testing NLP models. *CheckList* is a list of general linguistic capabilities and test type baselines for NLP tasks. It is also a software tool to generate test cases for NLP models.

2.3 Behavioral Studies in Knowledge Tracing

The expected behaviors of the KT models have been discussed in some studies, which point out the adversarial behaviors of KT models and propose new models to alleviate the problem. DKT+ [41] raises two problems of the Deep Knowledge Tracing (DKT) model [31], which are increasing correctness probabilities from false responses, and wavy prediction transition by time. However, these behaviors can naturally occur from the educational effects embedded in the interaction, which we discuss in detail in Section 3.1. The authors add three regularization terms in DKT+ to enhance the consistency of the predictions of DKT, and introduce extra performance measures.

The authors of [19] lists some desirable behaviors based on the monotonicity of the KT models to improve the general ability of the models. Then, they perform three types of novel data augmentation techniques(replacement, insertion, and deletion) and apply them to the training of KT models.

As examined in these relevant studies, the adversarial behaviors and low interpretability of DL models hinders the AIEd society to adopt Deep Learning based KT (DLKT) models and sustain on adopting interpretable models based on BKT, IRT, or Cognitive Diagnosis Models [9]. In this study, we provide a validation framework of DLKT models and conduct an extensive set of experiments on the desired behaviors of DLKT models. Good results highlight the reliability of DLKT models and encourages the model's adoption on most datsets. On the other hand, bad results point out the limitations of DLKT models on some datasets and show spaces for improvement.

3. BEHAVIORAL TESTING FOR KNOWLEDGE TRACING

We propose a black-box behavioral testing framework for knowledge tracing task. First we define the knowledge state (KS), and then elaborate on desirable behaviors of KT models' KS representation. Finally, in Subsection 3.2, we introduce specific experiment setups to assess whether DLKT models satisfy those behaviors.

We define the ***knowledge state*** to be a vector representation of a user's correctness probability on a set of questions Q' at a specific time point. Given the first t interactions $X_t = x_1, x_2, \cdots, x_t$ of a user, we define the user's knowledge state as:

$$KS_t = \left[P(c_t^{(q_j)} = 1 | X_{t-1}, q_j)\right]_{q_j \in Q'} \quad (1)$$

$c_t^{(q_j)}$ represents the Bernoulli indicator for the event when the user answers correctly to question q_j at time step t, as defined in Table in Section 2.1, which is updated along the provision of the user interaction sequence. Note that a KS is the collection of prediction values of questions, which either is responded or not. We describe the desired aspects of DLKT models in the following section.

3.1 Expected Behaviors of Traced Knowledge State

First we introduce two properties on the ***change*** ΔKS of the knowledge state KS with respect to an atomic change ΔX of the interaction sequence X, which is an insertion or a deletion of single interaction record.

First, **monotonicity** insists that the model's knowledge state should be updated to a more knowledgeable state when the student adds another correctly answered question (positive interaction) or when an interaction record with incorrect response (negative interaction) of the student is deleted. If the Δ is applied in the middle of the interaction record, all changes after the perturbation should hold the property as well. Second, **robustness** insists that a little perturbation in the interaction history should not yield a dramatic change in future knowledge states. The details of the two properties are introduced below.

- **Monotonicity**: If ΔX is a correctly responded interaction $(q_{i_{t_p}}, \mathbf{1})$ at perturbation time t_p, then we can track the relation of $P(c_t^{(q_j)} = 1 | X \cup \Delta X, q_j)$ and $P(c_t^{(q_j)} = 1 | X, q_j)$ depending on how q_j and ΔX are correlated. In many cases, a positive correlation in correctness probabilities is desired due to the relation of knowledge states:

  $$P(c_t^{(q_j)} = 1 | X \cup \Delta X, q_j) > P(c_t^{(q_j)} = 1 | X, q_j)$$

  for $t > t_p$ and $q_j \in Q$.

  However, there can also be negatively correlated questions which could be consequences of factors such as limited learning resource. For instance, a college student might sacrifice her studying time on one subject over another when both subjects' examinations are scheduled too closely with each other. This type of circumstance might cause the model to fit a non-monotonic relationship between the two fundamentally unrelated subjects. In most ITS's, however, the target study domain is usually restricted to a single subject, or a set of knowledge components where the student's comprehension on the components is usually positively correlated. Another case is when a negative response increases the correctness probability of a problem as described as an adversarial behavior in [41]. However, the educational effect of consuming a question can give positive feedback on the knowledge state even if the interaction response was wrong. Therefore, we assume the described monotonic behavior in general knowledge tracing environments while simultaneously keeping track of the opposite case in the experiments of Section 4.

- **Robustness**: For any black-box system, it is generally desirable that insignificant change in the system's input leads to limited change in the output. For a knowledge tracing model in an ITS, the input refers to the student's interaction record and the output refers to the model prediction on correctness probability for an encountered question or a set of questions. Therefore, we formulate the robustness of knowledge tracing model as below in a general sense, adopting the ΔX previously defined:

  $$|P(c_t^{(q_j)} = 1 | X \cup \Delta X, q_j) - P(c_t^{(q_j)} = 1 | X, q_j)| < \epsilon_t$$

  for some ϵ_t, a single interaction ΔX, $t > t_p$, and $q_j \in Q$. If we impose the inequality to always hold on $t = t_p + 1$ and fixed ϵ_1, then it is equivalent to imposing **continuity** on the knowledge state in terms of time-steps. We treat continuity as a specific case of **robustness** and introduce customized test for continuity separately from the test for robustness.

  However, consider a case when q_j and $q_{i_{t_p}}$ in ΔX assess similar concepts, or when the educational effect from the interaction with one question affects the student's correctness probability on the other question. Then an insertion or deletion of one question is prone to have a significant impact upon the predicted correctness probability value of the other for $t > t_p$. Therefore, the defined robustness / continuity need not be universally desirable for all pairs of questions. The impact of this property would eventually depend on the degree of dependency among the questions. Therefore, in the experiments, while assuming robustness for most question pairs, we also carefully track where some questions affect the prediction values of other questions in a notable amount.

Next we discuss what constitutes an expected ***value*** of knowledge state. Testing whether the knowledge state has accurately captured the user's interaction history is in line with the existing quantitative metrics (AUC, ACC) adopted in KT literature. However, the existing test methods focus only on a **single actual question** data provided per each time step whereas we propose to assess knowledge tracing model via its knowledge state on a **virtual question set** in order to provide a more holistic assessment via knowledge state representation.

Although tracing knowledge state on a set of questions would provide a more comprehensive picture of how the user's knowledge is traced, it lacks actual label of correctness on unseen questions at each time step, as discussed in Section 3.1. Therefore, we describe below novel measures to assess correctness of knowledge state under a few purposefully designed circumstances.

- Approximate Label of User-Independent **Initial Knowledge State**: At first prediction step, we approximate correctness label for *all* questions via their 'global difficulty'. The initial knowledge state for all users should represent the model's prior belief of question difficulty before encountering any user-specific interaction record data. It is reasonable to assess the quality of this value in terms of correlation of model's prediction on each question and the question's global difficulty. However, this is an approximation at best since the question's difficulty might not be accurately captured by simple average over its occurrences. If the actual interaction data generated from the ITS provides a very difficult question only after user's knowledge is significantly accumulated, simple average of question correctness labels would not be representative of the question's inherent difficulty. Model's prediction would be high.
- Ideal Value of Knowledge State after **Converged Interaction Data**: We generate obvious edge-case test cases where user's knowledge state on a set of question has converged to a value. We create this virtual dataset by simply repeating an identical interaction record on each question consecutively. The model's prediction value for the question in the repeated interaction should converge to the repeatedly provided label value.
- Approximate Label of Knowledge State in General: It's also possible to approximate a pseudo-label for unseen questions using rolling/expanding averages or IRT-like algorithms which demonstrate more stable and monotonic behavior by design. Although we conjecture such training methodology of DLKT models using pseudo-labels might provide regularization effect, we do *not* include this case in the scope of this work.

Table 1: Behavioral Test Summary

Behavior	Analysis Method
Monotonicity	**Perturbation Test**: Percentage of interaction samples of which model prediction changed in expected direction.
Robustness	**Perturbation Test**: Degree of impact from perturbation across time-steps.
Continuity	**Continuity Test**: Avgerage and maximum change in knowledge state score per step and throughout entire sequence.
Initial Value	**Initial Value Test**: Correlation between question correctness rate and initial knowledge state.
Convergence	**Convergence Test**: Convergence speed as in model AUC and average model output at different time-steps.

3.2 Behavioral Test Setups

Below we describe four behavioral testing setups for DLKT models. First, perturbation tests aim to test model's monotonicity and robustness given an atomic perturbation to the original interaction sequence data. Second, continuity test aims to check whether model's knowledge state representation is continuous along the interaction sequence. Third, initial knowledge state test checks whether the initial knowledge state reflects each question's corresponding difficulty measure. Fourth, convergence test checks whether the knowledge state converges to the expected value and how fast it converges. Following subsections elaborate each of the test setup in further detail. Table 1 provides summary of the tests.

3.2.1 Perturbation Tests

We examine monotonicity and robustness of the model by perturbation tests. We experiment three types of perturbations: insertion, deletion, and flip. For each original interaction sequence, we determine t_p, which is the index of interaction to be perturbed. For the insertion case, we add a new interaction between x_{t_p-1} and x_{t_p}. For the deletion case, we remove the interaction x_{t_p}. For the flip case, we flip the correctness of x_{t_p} from 1 to 0 and from 1 to 0.

In order to check monotonicity, we assess whether the model's predicted correctness probability in the following interaction sequence $X_{[t_p:]}$ changes towards the expected direction. For insertion / deletion / flip of an interaction to which user responded correctly, we examine whether the following future correctness probability $P(c_{t'+1} = 1|X_{t'})$, $\forall t' > t_p$ increases / decreases / decreases, respectively. In the experiments, we fix the perturbation point to be located halfway in the user's original interaction sequence, then measure the proportion of interactions which the model's predicted correctness probability changes towards the expected direction.

To assess the model's robustness, we visualize how the degree of impact from perturbation changes along the time steps from t_p. We expect the degree of impact from perturbation upon the model's prediction to decay gradually as more interactions are fed into the model after the perturbation point t_p.

3.2.2 Continuity Test

We test whether the knowledge state is continuous, in the manner described in the previous section 3.1. For every time-step, we provide the model with not only the original interaction at the corresponding time-step, but also a set of questions Q' simultaneously to construct knowledge state KS_t at the time-step. Although we don't have actual correctness label for those virtual interactions, we only inquire how the knowledge state or the model prediction on Q' evolves along the time-steps.

In the experiments, we approximate a **score** on the user's knowledge state by averaging the model-predicted correctness probability over the sample set of questions Q' to report: (1) average and maximum student score change per single time-step and (2) average student score change and range across 100 time-steps.

3.2.3 Initial Knowledge State Test

Table 2: Dataset Statistics

Dataset	Users	Items	Skills	#Intr.	%Crct.
ASSIST15	14228	100	100	656K	73
ASSIST17	1708	3162	411	935K	37
STATICS	282	1223	98	189K	77
Spanish	182	409	221	579K	77
EdNet-small	5000	13156	118	518K	65
EdNet-med	100000	13518	118	11M	64
EdNet	605763	13528	118	138M	66

We assess the quality of the prior knowledge state embedded by the model by the initial knowledge state test. Without any user-specific record, the prior knowledge state embedded in the model should accurately reflect the average difficulty of the question to all users. Thus, we check Spearman rank correlation and Pearson correlation between the question's average difficulty and the model-predicted prior belief for each question.

In detail, a trained DLKT model M's initial knowledge state for a question q_j can be represented as $P_M(c = 1|\cdot, q_j)$. We compare this with the question correctness rate over the entire dataset as in Eq 2 which is equivalent to the number of correctly responded q_j-interactions over the number of occurrences of q_j based on all user data.

$$\mathrm{gc}_{q_j} = \frac{\sum_{u \in U} |\{x_t^{(u)} | q_{i_t} = q_j, c_t = 1\}|}{\sum_{u \in U} |\{x_t^{(u)} | q_{i_t} = q_j\}|} \tag{2}$$

Consequently, we measure:

$$Corr\Big(\big[P_M(c = 1|\cdot, q_j)\big]_{q_j \in Q} \ , \ \big[\mathrm{gc}_{q_j}\big]_{q_j \in Q} \Big) \tag{3}$$

This initial knowledge state test pinpoints on whether the learned question embedding in the DLKT model alone has captured any information about the corresponding question's difficulty. Moreover, we emphasize the importance of the initial knowledge state since the state assumed by the model would likely affect the user's first impression on the system to make decisions.

3.2.4 Convergence Test

In convergence test, we assess whether the model's knowledge state value converges to a target value in a desired manner. We generate simple virtual interaction sequence data by repeating an identical interaction for 50 time-steps for each question for both correctness cases. Therefore, the virtual dataset would consist of virtual user interaction sequences of size twice of the number of questions.

In the experiments, we report the model's standard AUC metric at time-steps 5, 10, and 50. We expect significantly high figures as the inquired interaction sequence is extremely simple. We also visualize how the average model prediction value across the questions evolves throughout the 50 time-steps for each of the correctness case. We expect the values to quickly converge to 1 / 0 for interaction sequences of which correctness label is all correct / incorrect, respectively.

4. EXPERIMENTS

In this section, we benchmark three Deep Learning based Knowledge Tracing models DKT, SAKT, and SAINT on the proposed behavioral tests. First, we train optimized models for each architecture-dataset pair by searching hyper-parameters on the train and validation data split. Second, we report the classification accuracy and the AUC metric, which are commonly used for model assessments in the Knowledge Tracing literature. Third, we present the proposed behavioral test results of model instances on well-known datasets for Knowledge Tracing.

4.1 Datasets

We describe the datasets used in our experiments. All datasets are open to the public.

ASSISTments[11] is a dataset containing student interactions from an online tutoring system for solving Massachusetts Comprehensive Assessment System (MCAS) 8th Math test questions. We use the datasets ASSISTments 2015 (**Assistments15**) and ASSISTments Challenge 2017 (**Assistments17**).

STATICS is a dataset containing college student interactions on a one-semester Statics course. This dataset is available in the PSLC DataShop web site [16].

Spanish[20] is a dataset containing middle-school student interaction data for Spanish exercises.

EdNet[6] is the largest public benchmark education dataset containing user interaction data of an online tutoring system, for preparing TOEIC (Test of English for International Communication®). For ablation studies on the size of the dataset, we randomly choose 100,000 users for **EdNet-medium**, and 5,000 users for **EdNet-small**.

Table 3: Model Hyper-parameters

Model	Parameter	Tuning Details
Common	Adam learning rate	0.001, 0.003, 0.01
	Dropout rate	0, 0.25, 0.5
	Embedding dimension	64, 128, 256
	Maximum Seq.Length	100, 200, 400
DKT	# Recurrent Layers	1, 2, 4
SAKT	# Attention Layers	1, 2, 4
SAINT	Warm-up Steps	200, 400, 4000
	# Attention Head	1,4,8

4.2 Models and Algorithms

We perform hyper-parameter tuning on the training of models. For each configuration of hyper-parameters, we choose the model weights with the best validation AUC. In the training step, an early-stopping policy is applied with patience 30, which means that we stop the training process and save the best weights if there is no AUC improvement in the recent 30 validation steps. Among the best weights for each configuration, we choose the weight with the best validation AUC for each dataset, and evaluate the weights with an independent test set for test metrics.

4.2.1 Training Details

In this study, we use DKT, SAKT, SAINT in the experiments . DKT models the student's knowledge state using a Recurrent Neural Network (RNN) by compressing the interaction history in a hidden layer. SAKT is the first KT model that used self-attention layers, where in each layer the question embeddings are queries and interactions embeddings are key and values. SAINT is the first KT model based on Transformers. The sequence of questions is fed into the encoder, and the sequence of responses are fed into the decoder with the encoder output.

Our model hyper-parameters are shown in Table 3. We use the Adam optimizer [15] with default parameters. For SAKT and SAINT, we used the Noam scheme for scheduling the learning rate, and tune the number of warmup-steps.

The original SAKT implementation does not include residual connection from the query. This enforces the first prediction to a same number every time, regardless of the first question provided to the model. Since 3.2.3 becomes redundant, we use the modified SAKT architecture with residual connection. For SAINT and SAKT, the dimension of the feedforward network is set to 4×(model dimension). For SAINT, we use the same number of attention layers for the encoder and the decoder.

4.3 Results: Traditional Assessment

AUC and accuracy results are shown in Tables 4 and 5. The difference of these standard metrics is generally less than 0.01. For KT-based tutoring systems, this difference would be less important than behavioral performance. AUC shows the monotonicity of interactions by all users, and accuracy does not focus on the exact model prediction. On the other hand, behavioral tests can check the performance of model prediction for a single user, and analyze the impact of a single interaction.

Table 4: Standard AUC metric

Model	DKT	SAKT	SAINT
Assistments15	0.7242	0.7226	0.7179
Assistments17	0.7742	0.7650	0.7680
STATICS	0.8269	0.8248	0.8275
Spanish	0.8336	0.8456	0.8364
EdNet-small	0.7332	0.7380	0.7328
EdNet-medium	0.7717	0.7760	0.7722
EdNet	0.7810	0.7905	0.7863
Average	0.7778	0.7804	0.7773

Table 5: Standard Classification Accuracy(%)

Model	DKT	SAKT	SAINT
Assistments15	74.2	74.6	74.4
Assistments17	72.1	71.0	71.8
STATICS	81.4	81.2	81.1
Spanish	81.9	82.6	82.0
EdNet-small	68.2	70.2	69.8
EdNet-medium	72.5	72.6	72.4
EdNet	73.5	74.1	73.9
Average	74.8	75.2	75.1

4.4 Results: Behavioral Testing

4.4.1 Perturbation Tests

We report the test pass rate for insertion, deletion, and flip. The results are shown in Table 6, 7, and 8, respectively. Figure 1 describes the average impact on model prediction from insertion perturbation on each dataset (column) and correctness label of the inserted interaction (row). Figure 2 describes the degree of maximum impact over user sequences from insertion perturbation.

Table 6: Insertion Test Pass Rates(%)

Model	DKT	SAKT	SAINT
Assistments15	70.9	70.3	65.3
Assistments17	69.6	55.7	56.7
STATICS	71.1	61.0	58.2
Spanish	80.1	75.6	60.7
EdNet-small	66.3	78.0	75.9
EdNet-medium	83.2	80.6	77.3
EdNet	72.7	71.6	71.2
Average	73.4	70.4	66.5

Table 7: Deletion Test Pass Rates(%)

Model	DKT	SAKT	SAINT
Assistments15	69.0	66.3	62.1
Assistments17	63.7	54.4	54.6
STATICS	60.7	55.9	49.2
Spanish	81.6	81.9	59.7
EdNet-small	65.6	75.3	71.9
EdNet-medium	80.3	76.8	73.5
EdNet	72.3	68.3	69.5
Average	70.4	68.4	62.9

Table 8: Flip Test Pass Rates(%)

Model	DKT	SAKT	SAINT
Assistments15	77.1	96.3	94.7
Assistments17	69.5	86.1	66.4
STATICS	93.4	92.9	84.7
Spanish	87.5	89.1	83.9
EdNet-small	75.2	95.0	95.8
EdNet-medium	87.8	94.8	95.5
EdNet	79.5	83.6	86.0
Average	81.4	91.1	86.7

- In general, deletion and insertion pass rates range from 60% to 80%, and flip pass rates range from 80% to 90%. Note that a flip can be interpreted as a combination of deletion and insertion. Therefore, the impact of perturbation is supposed to be larger, leading to higher pass rates as compared to insertion/deletion cases. From Figure 1, Figure 6 (Appendix), and Figure 7 (Appendix), we note that the degree of impact from replacement is twice of that from insertion or deletion.
- Robustness: From Figure 1, we observe that the degree of average impact from perturbation gradually decreases along the time-steps in general, and that the average impact is limited by only about 2%. Therefore, the desired robustness holds in terms of average impact.
- Monotonicity: From Figure 1, the average impact from positive/negative perturbation tends to remain posi-

Figure 1: Perturbation Test: Average Impact on Model Prediction from Insertion.

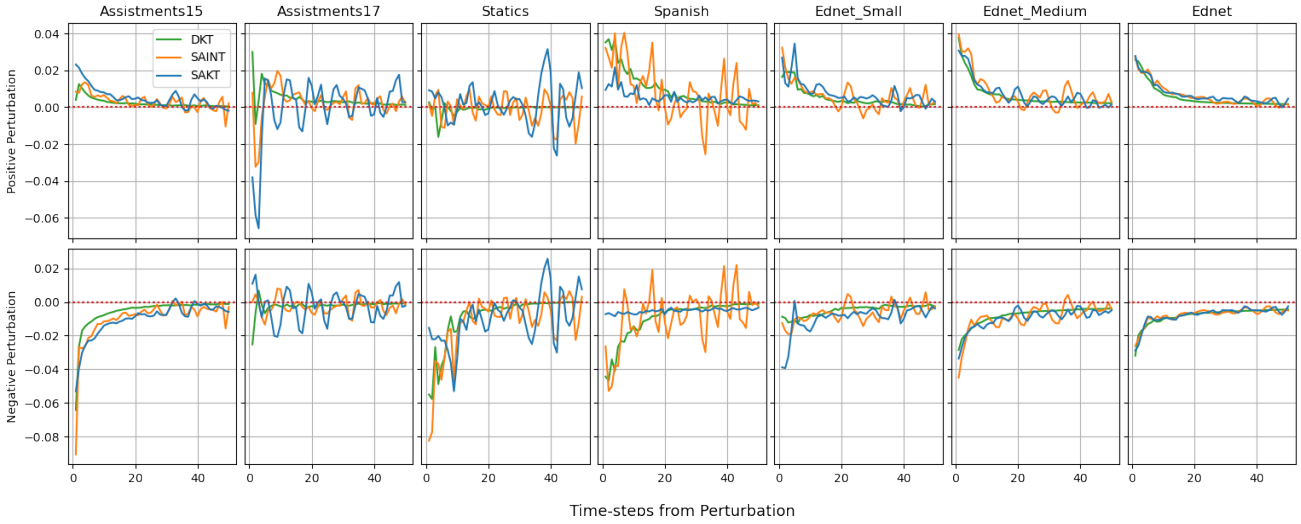
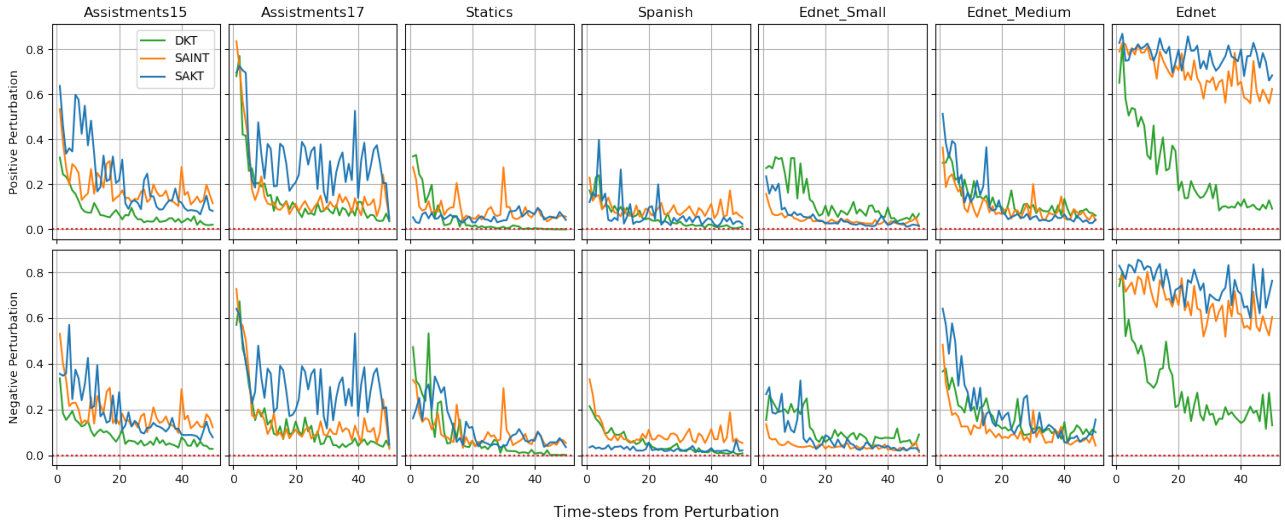


Figure 2: Perturbation Test: Maximum Degree of Impact on Model Prediction from Insertion.



tive/negative, respectively, for Assistments15, EdNet-small, EdNet-medium, and EdNet datasets. On Spanish dataset, such trend is more noisy for SAINT network.

- On Assistments17 and Statics dataset, the expected monotonic behavior from SAKT and SAINT is not observed as the average impact oscillates across zero. This can be also seen from DKT's significantly higher pass rates on the two datasets in Table 6.
- From Figure 2, we note that there exists questions which persist to respond in a larger degree (even up to 80%) after 40 time-steps. Note that on the EdNet dataset, both transformer-based architectures SAKT and SAINT allow larger impacts from perturbations than DKT. This can be explained by the superior performance of the two models on EdNet data over DKT in terms of standard evaluation metrics AUC and ACC.

4.4.2 Continuity Test

We report average and maximum step-wise change in KS score over students in Table 9. Apart from the single-step change, we also measure final change of score from the first time-step to the last, and the total range of score explored throughout the time-steps, averaged over all students in Table 10. Sum of absolute change in KS coordinates, or Manhattan distance of KS's along time-steps (averaged over all students) is shown in Figure 3. EdNet-medium was omitted due to its similarity with the plot from EdNet-small.

- Except for Assistments17 and Statics, average score change per single time-step or an interaction remains reasonably low below 5% for all architectures. This suggests that the knowledge state is fairly stable across the time-steps.
- On Assistments17 and Statics, we observe significantly larger changes, especially in DKT. DKT's maximum

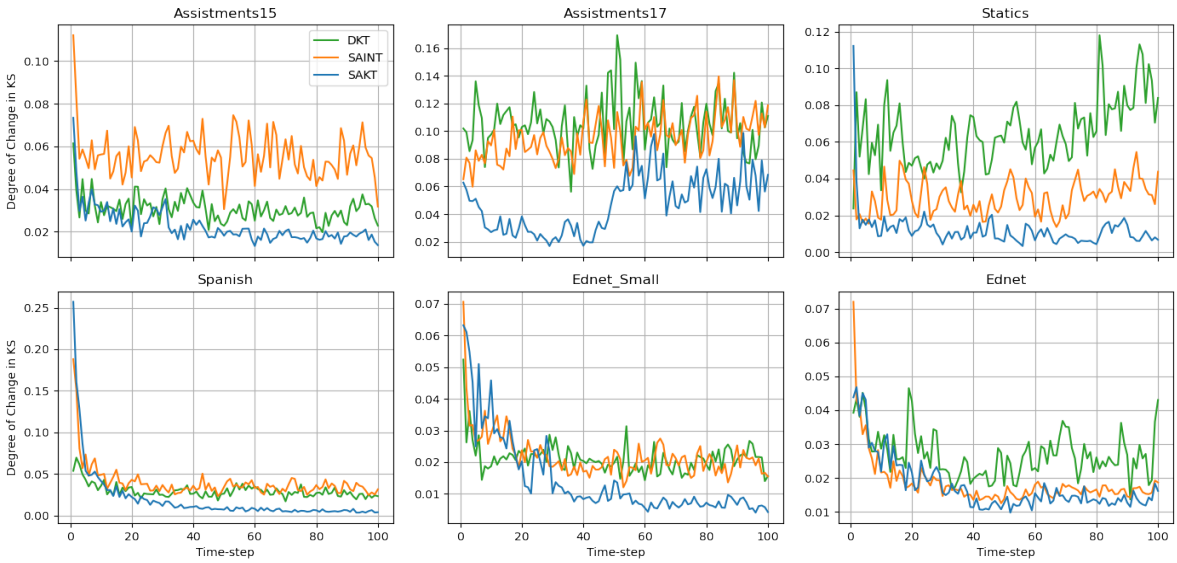


Figure 3: Continuity Test: Average Change in KS along Time-steps

Table 9: Continuity Test: Average / Maximum Student Score Change(%) per Single Time Step

Model		Assist15	Assist17	STATICS	Spanish	EdNet-small	EdNet-med	EdNet	Average
DKT	Avg	2.45	10.48	6.39	2.93	2.04	2.14	2.27	4.10
	Max	16.82	84.97	65.09	20.33	16.98	17.06	33.68	36.42
SAKT	Avg	1.97	3.92	1.15	1.68	1.29	1.19	1.48	1.81
	Max	15.65	56.60	20.24	52.92	21.46	19.21	22.53	29.80
SAINT	Avg	4.64	7.99	2.73	3.48	2.02	1.92	1.29	3.44
	Max	31.80	53.01	24.18	38.74	16.17	22.12	17.19	29.03

Table 10: Continuity Test: Average Student Score Total Change(%) / Total Range(%) over 100 Interactions

Model		Assist15	Assist17	STATICS	Spanish	EdNet-small	EdNet-med	EdNet	Average
DKT	Diff	10.14	11.50	9.94	18.30	9.23	12.82	10.83	11.82
	Range	24.72	63.42	47.54	46.54	22.89	27.41	28.37	37.27
SAKT	Diff	15.96	13.97	15.67	18.36	10.48	13.00	8.95	13.77
	Range	30.98	40.16	22.08	53.33	23.70	23.80	20.69	30.68
SAINT	Diff	13.34	11.62	6.97	20.53	11.32	5.30	9.01	11.15
	Range	37.98	49.96	26.19	51.51	22.98	24.21	20.59	33.35

score change across a single time-step is as high as 85% and 65% for Assistments17 and Statics, respectively.

- In general, we observe decreasing marginal impact of each interaction data as time proceeds.
- From Figure 3 and Table 9, we note that SAKT's knowledge state changes significantly less than other models, consistently throughout all datasets. We also investigated whether this 'speed' of change affects total 'dislocation' of knowledge state in Table 10. Interestingly, SAKT's knowledge state moved by farthest on average (13.77%) while its range explored was the smallest (30.68%) on average. This suggests that SAKT's knowledge state evolution was least volatile.

4.4.3 *Initial Knowledge State Test*

To assess the validity of initial knowledge states embedded in the model, we measured the correlation of the predicted prior knowledge state and the global question difficulty as described in Section 3.2.3. The results are presented in Table 11. In the scatter plot of Figure 4, we choose the 200 most frequently answered questions from each data-set to show how the initial model predictions and question correctness rates are distributed and correlated.

- We observe from Table 11 that all models' initial knowledge states are positively correlated with the global question difficulty with statistical significance.
- The difference in correlation metrics among datasets is much more significant than that among models.
- Based on the three scatter plots of the first row in the figure, we note that the correlation becomes stronger as the size of dataset grows from EdNet_Small to full EdNet data. Table 11 and Table 2 also suggests that the number of interactions per unique question is positively correlated with the initial knowledge state test metric.

- From the scatter plot, we see that the three models occupy slightly different clustering regions in the plot. For instance, in EdNet_Medium and EdNet dataset, SAINT's initial prediction value is consistently larger than that of the other two models, which suggests ensemble of the models to reduce bias.

Table 11: Initial Knowledege State Test: Correlation(%)

Model	DKT	SAKT	SAINT
Assistments15	85.6	84.8	82.5
Assistments17	63.2	52.2	58.2
STATICS	56.1	58.1	54.6
Spanish	56.5	49.1	44.0
EdNet-small	39.0	38.5	31.6
EdNet-medium	75.1	74.2	74.4
EdNet	87.6	86.5	88.2
Average	66.1	63.3	61.9

4.4.4 Convergence Test

As the dataset we generate and use for the convergence test is extremely simple as described in Section 3.2.4, we expect the KT models' standard performance metrics to increase quickly along the time-steps. For instance, at the fifth time-step, the model would have already received four equivalent interaction record with the same question and the same correctness label for the virtual student. We report the model AUC at time-step 5, 10, and 50 in Table 12. In Figure 3 we also visualize the model's average response across different questions for each correctness label values assumed. We expect the average response plot to quickly converge to either 1 or 0 based on the assumed correctness label value.

- In general, all models show fairly high performance from early time-step of 5, except for Assistments17 dataset.
- Both Table 12 and Figure 5 suggest SAKT consistently achieves fastest convergence to a reasonable value close enough to either 1 or 0. DKT, however, consistently converges to a value farther from the two edges, as compared to the other two models. In particular, for the incorrect case (second row) of the Figure 5, we observe DKT converges to a value higher than 50% (red dotted horizontal line). For the positive case (first row), DKT converges to a correctness probability level around only 70% for Assitments17, EdNet-small, and EdNet-medium.
- DKT's convergence pattern is fairly monotonic while SAKT and SAINT's patterns go through fluctuation which likely pertains to noise.
- It is noteworthy that increasing dataset size from EdNet-small to EdNet-medium and EdNet significantly helps all three models' convergence behavior on both target correctness values, especially for DKT. DKT's convergence value moved signficantly closer to desired values of 1 and 0. For SAKT and SAINT, larger dataset size led to more stable response plot, reducing the degree fluctuation.
- Convergence in the incorrect case and the correct case is asymmetric. While the latter closely achieves the target value of 1, the former case converges around 30% level in most datasets. We attribute this to the tutorial content embedded in each of the interaction, along with the question item used for assessment in the dataset.

4.5 Overview of Experimental Results

Based on the proposed DLKT validation framework, we conducted a comprehensive investigation of three popular DLKT models on seven benchmark datasets to scrutinize the models' behavioral characteristics. The results highlight strengths and weaknesses of three DLKT models. Although DLKT models demonstrated stable and robust behaviors in line with expectation in most datasets, the results revealed few major disadvantages for each models: DKT showed better stability in perturbation tests while the other architectures occasionally presented volatile fluctuation in the response curve. In the continuity test, SAKT presented a significantly smoother evolution of knowledge state, but other models' knowledge state representations were seemingly volatile in a few datasets which strongly precludes DLKT's adoption. On the other hand, this suggests room for improving DLKT models based on the specific issue pinpointed by this framework. For instance, the volatility of KS could be alleviated by direct regularization of the change in the KS. On the other hand, the results from the convergence test showed that DKT was fragile even to simple edge-case data which undermines generalization capability of DKT, as compared to other attention-based architectures.

These behavioral characteristics identified from the proposed framework show that the two popular architectural paradigms, RNN and Attention-based, possess different strengths and weaknesses under KT environment. This also hints that an architectural combination or ensemble approach might alleviate the identified issues to improve both standard KT model evaluation metrics and behavioral characteristic.

5. CONCLUSION

In this work, we introduced the desired properties of knowledge tracing models and proposed a novel model validation framework for Deep Learning based Knowledge Tracing (DLKT) models. Using the framework, we conducted a comprehensive analysis of three popular DLKT models' behavioral characteristics and identify their strengths and weaknesses of the models in seven different benchmark datasets. We believe that the analysis on both strengths and weaknesses diagnosed by the framework would serve as a useful guideline for model enhancement. Also based on the findings from the proposed framework, a customized adoption of DLKT models fitting to the nature of the data and desired behaviors as well as accuracy would become possible.

We believe potential future work includes: (1) tackling the weaknesses of DLKT models identified in this work via architectural modification or model combination, (2) exploring the benefit of data augmentation using virtual edge-case data similar to converging interaction data used in the convergence test, and (3) extending the proposed testing framework beyond the task of knowledge tracing (i.e. student score prediction and item recommendation).

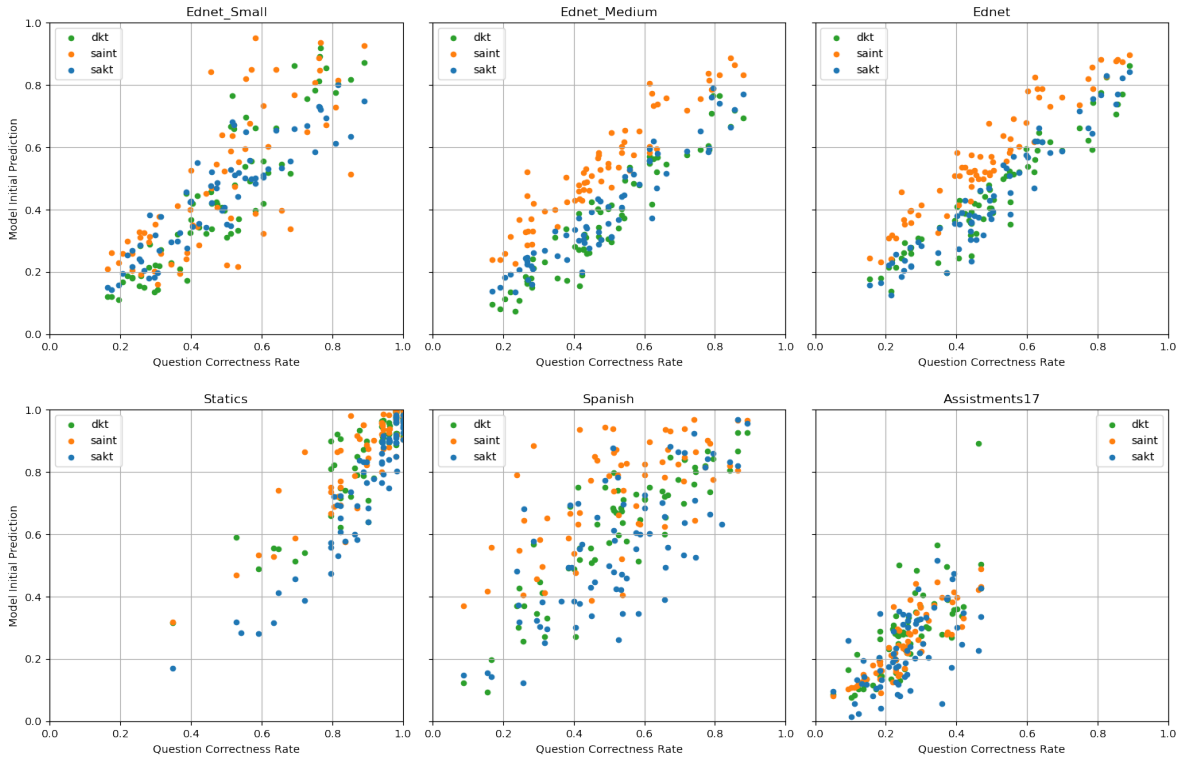


Figure 4: Initial Knowledge State Test Scatter Plot

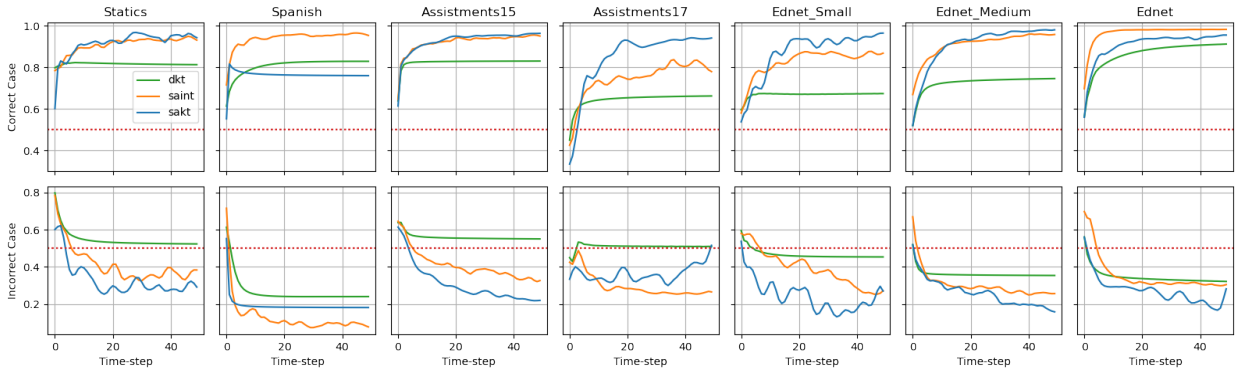


Figure 5: Convergence Test: Average Model Prediction along Converging Interaction Sequence

Table 12: Convergence Test AUC

Step	Model	Assist15	Assist17	STATICS	Spanish	EdNet-small	EdNet-med	EdNet	Average
5	DKT	0.867	0.601	0.829	0.688	0.622	0.790	0.845	0.749
	SAKT	0.885	0.615	0.903	0.805	0.869	0.853	0.861	0.827
	SAINT	0.866	0.609	0.936	0.731	0.628	0.881	0.854	0.786
10	DKT	0.932	0.634	0.924	0.745	0.679	0.874	0.932	0.817
	SAKT	0.961	0.782	0.928	0.923	0.953	0.928	0.951	0.918
	SAINT	0.947	0.763	0.979	0.856	0.757	0.958	0.954	0.888
50	DKT	0.979	0.695	0.983	0.791	0.744	0.954	0.990	0.876
	SAKT	0.998	0.938	0.942	0.993	0.997	0.994	0.995	0.980
	SAINT	0.995	0.939	0.999	0.977	0.963	0.997	0.997	0.981

6. REFERENCES

[1] G. Abdelrahman and Q. Wang. Knowledge tracing with sequential key-value memory networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 175–184, 2019.

[2] B. Beizer. *Black-box testing: techniques for functional testing of software and systems*. John Wiley & Sons, Inc., 1995.

[3] C. G. Brinton and M. Chiang. Mooc performance prediction via clickstream data and social learning networks. In *2015 IEEE conference on computer communications (INFOCOM)*, pages 2299–2307. IEEE, 2015.

[4] P. Chen, Y. Lu, V. W. Zheng, and Y. Pian. Prerequisite-driven deep knowledge tracing. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 39–48. IEEE, 2018.

[5] Y. Choi, Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and J. Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pages 341–344, 2020.

[6] Y. Choi, Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, B. Kim, and Y. Jang. Ednet: A large-scale hierarchical dataset in education, 2019.

[7] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[8] R. S. d Baker, A. T. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International conference on intelligent tutoring systems*, pages 406–415. Springer, 2008.

[9] J. De La Torre and J. A. Douglas. Higher-order latent trait models for cognitive diagnosis. *Psychometrika*, 69(3):333–353, 2004.

[10] S. E. Embretson and S. P. Reise. *Item response theory*. Psychology Press, 2013.

[11] M. Feng, N. Heffernan, and K. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User modeling and user-adapted interaction*, 19(3):243–266, 2009.

[12] T. Gervet, K. Koedinger, J. Schneider, T. Mitchell, et al. When is deep learning the best approach to knowledge tracing? *JEDM| Journal of Educational Data Mining*, 12(3):31–54, 2020.

[13] A. Ghosh, N. Heffernan, and A. S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2330–2339, 2020.

[14] F. Gutierrez and J. Atkinson. Adaptive feedback selection for intelligent tutoring systems. *Expert Systems with Applications*, 38(5):6146–6152, 2011.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the edm community: The pslc datashop. *Handbook of educational data mining*, 43:43–56, 2010.

[17] E. R. Lai. Metacognition: A literature review. *Always learning: Pearson research report*, 24:1–40, 2011.

[18] M. Last, S. Eyal, and A. Kandel. Effective black-box testing with genetic algorithms. In *Haifa Verification Conference*, pages 134–148. Springer, 2005.

[19] S. Lee, Y. Choi, J. Park, B. Kim, and J. Shin. Consistency and monotonicity regularization for neural knowledge tracing, 2021.

[20] R. V. Lindsey, J. D. Shroyer, H. Pashler, and M. C. Mozer. Improving students' long-term knowledge retention through personalized review. *Psychological science*, 25(3):639–647, 2014.

[21] Q. Liu, Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, and G. Hu. Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, 33(1):100–115, 2019.

[22] Y. Liu, Y. Yang, X. Chen, J. Shen, H. Zhang, and Y. Yu. Improving knowledge tracing via pre-training question embeddings. *arXiv preprint arXiv:2012.05031*, 2020.

[23] E. Loken and K. L. Rulison. Estimation of a four-parameter item response theory model. *British Journal of Mathematical and Statistical Psychology*, 63(3):509–525, 2010.

[24] M. I. Lopez, J. M. Luna, C. Romero, and S. Ventura. Classification via clustering for predicting final marks based on student participation in forums. *International Educational Data Mining Society*, 2012.

[25] Y. K. Malaiya. Antirandom testing: Getting the most out of black-box testing. In *Proceedings of Sixth International Symposium on Software Reliability Engineering. ISSRE'95*, pages 86–95. IEEE, 1995.

[26] L. Mariani, M. Pezze, O. Riganelli, and M. Santoro. Autoblacktest: Automatic black-box testing of interactive applications. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pages 81–90. IEEE, 2012.

[27] J. Martin and K. VanLehn. Student assessment using bayesian nets. *International Journal of Human-Computer Studies*, 42(6):575–591, 1995.

[28] H. Nakagawa, Y. Iwasawa, and Y. Matsuo. Graph-based knowledge tracing: modeling student proficiency using graph neural network. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 156–163. IEEE, 2019.

[29] S. Nidhra and J. Dondeti. Black box and white box testing techniques-a literature review. *International Journal of Embedded Systems and Applications (IJESA)*, 2(2):29–50, 2012.

[30] S. Pandey and G. Karypis. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*, 2019.

[31] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Advances in neural information processing systems*, pages 505–513, 2015.

[32] G. Rasch. *Probabilistic models for some intelligence and attainment tests.* ERIC, 1993.

[33] Z. Ren, X. Ning, A. Lan, and H. Rangwala. Grade prediction based on cumulative knowledge and co-taken courses. In *Proceedings of the 12th International Conference on Educational Data Mining (EDM)*. ERIC, 2019.

[34] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*, 2020.

[35] C. Romero, M.-I. López, J.-M. Luna, and S. Ventura. Predicting students' final performance from participation in on-line discussion forums. *Computers & Education*, 68:458–472, 2013.

[36] H. Tan, J. Guo, and Y. Li. E-learning recommendation system. In *2008 International Conference on Computer Science and Software Engineering*, volume 5, pages 430–433. IEEE, 2008.

[37] D. Thissen and M. Orlando. Item response theory for items scored in two categories. 2001.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[39] J.-J. Vie and H. Kashima. Knowledge tracing machines: Factorization machines for knowledge tracing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 750–757, 2019.

[40] T. Wang, F. Ma, and J. Gao. Deep hierarchical knowledge tracing. In *Proceedings of the 12th International Conference on Educational Data Mining*, 2019.

[41] C.-K. Yeung and D.-Y. Yeung. Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, pages 1–10, 2018.

[42] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*, pages 171–180. Springer, 2013.

[43] J. Zhang, X. Shi, I. King, and D.-Y. Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*, pages 765–774, 2017.

APPENDIX

Figure 6: Perturbation Test: Average Impact on Model Prediction from Deletion.

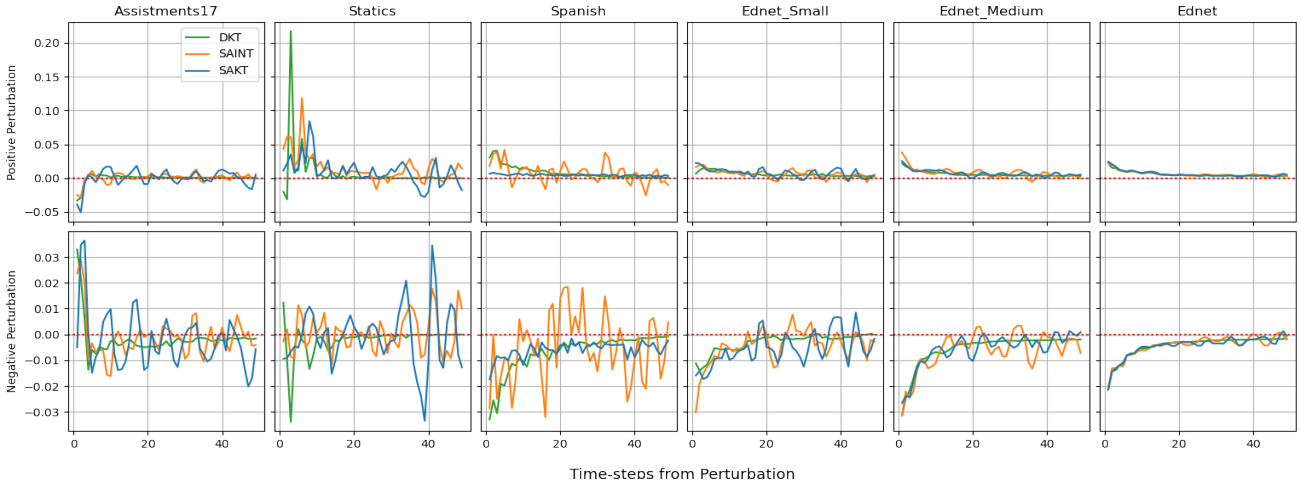
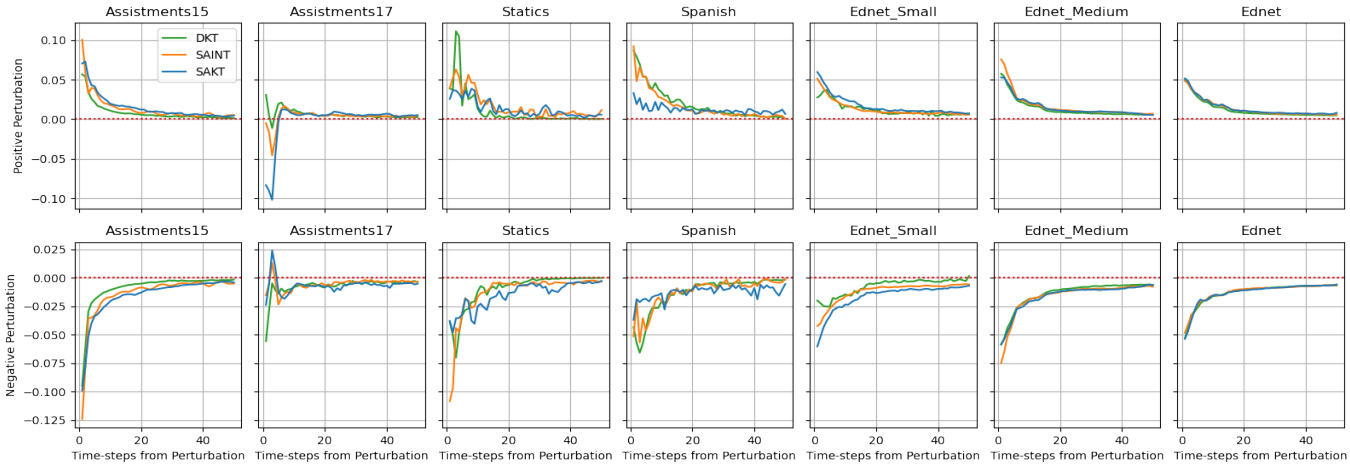


Figure 7: Perturbation Test: Average Impact on Model Prediction from Flip.



Student Strategy Prediction using a Neuro-Symbolic Approach

Anup Shakya
University of Memphis
ashakya@memphis.edu

Vasile Rus
University of Memphis
vrus@memphis.edu

Deepak Venugopal
University of Memphis
dvngopal@memphis.edu

ABSTRACT

Predicting student problem-solving strategies is a complex problem but one that can significantly impact automated instruction systems since they can adapt or personalize the system to suit the learner. While for small datasets, learning experts may be able to manually analyze data to infer student strategies, for large datasets, this approach is infeasible. We develop a Machine Learning model to predict strategies from student data. While Deep Neural Network (DNN) based methods such as LSTMs can be applied for this task, they often have long convergence times for large datasets and like several other DNN-based methods have the inherent problem of overfitting the data. To address these issues, we develop a Neuro-symbolic approach for strategy prediction, namely a model that combines strengths of symbolic AI (that can encode domain knowledge) with DNNs. Specifically, we encode relationships in the data using Markov Logic and use symmetries among these relationships to train an LSTM more efficiently. In particular, we use an importance sampling approach where we sample the training data such that for clusters/groups of symmetrical instances (instances where the strategies are likely to be symmetric), we only pick representative samples for training the model instead of using the whole group. Further, since some groups may contain more diverse strategies than the others, we adapt the importance weights based on previously observed samples. Through empirical evaluation on the KDD EDM challenge datasets, we show the scalability of our approach.

Keywords

Intelligent Tutoring Systems, Learning Strategies, Neuro-Symbolic AI, Markov Logic Networks, LSTMs

1. INTRODUCTION

Intelligent Tutoring Systems (ITSs) [31] and more broadly adaptive instructional systems (AISs)[1] help a diverse population of students by adapting instruction to each learner thus accounting for different learning abilities, learning styles and education goals. Such adaptation leads to more engaging and effective learning. However, in order to build effective ITSs, it is important to understand how students learn and what learning and instructional strategies are most effective for whom and under what conditions. Specifically, students can follow several different *strategies* to learn the same content. For example, consider a simple math problem about solving a linear system of equations where $x+y+z = 9$ $x = y$ and $y = z$. One strategy is to perform systematic substitutions till there is an equation in terms of one variable and simply solve for that variable. However, another strategy could be to use transitivity to see that all three variables have the same value and use this to solve the problem. Depending upon the way a student thinks, one strategy could be easier or harder to grasp compared to the other. Thus, understanding the various ways in which students approach an instructional task will not only further our understanding of how learners learn, e.g., it may help identify the most effective learning strategies employed by top performers, but it will also enable ITSs to incorporate knowledge about these strategies in order to adapt appropriately and help students maximize their learning gains.

A student's choice of strategy is a complex function dependent on many factors such as experience with similar problems, general expertise in the topic, other cognitive abilities, etc. Human experts are exploring all these factors and how they are related to strategy use and learning. However, human experts are expensive and limited in the ability to analyze large data from thousands, tens of thousands, or millions of students. Advanced data science methods and access to large computing infrastructure such as the cloud offer new possibilities to analyze in-depth and at scale such large learner datasets with the promise of helping us discover, document, and benefit from the diversity of learning.

Indeed, with the growth of both data and advanced Machine Learning methods such as Deep Neural Networks (DNNs), we are able to successfully solve several challenging problems in domains such as natural language understanding [25] and visual processing [8]. However, the ability of DNNs to learn complex functions and representations comes at a cost. Specifically, DNNs require significant computational

[1]The main difference between Intelligent Tutoring Systems and Adaptive Instructional Systems at least in our view is that the former offer full-adaptivity, i.e., both micro- and macro-adaptivity, whereas the latter can offer any type, e.g., just macro-adaptivity.

Anup Shakya, Vasile Rus and Deepak Venugopal "Student Strategy Prediction using a Neuro-Symbolic Approach". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 118-129. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

resources to scale up to large datasets and at the same time, as data increases, they may not always yield expected results since they have a tendency to *overfit* the data. That is, they work well on the data on which they were trained but their generalization performance on unseen data severely degrades. A paradigm that is gaining significant attention in the AI/Machine Learning research community is Neuro-symbolic AI [7] where we augment DNNs with symbolic models to regularize the DNN. This helps in improving both scalability and generalization by allowing DNNs to learn from smaller datasets with higher accuracy. In this paper, we apply a Neuro-symbolic model to predict student strategies from structured student-interaction data.

Our model works on student data where the student interacts with an ITS in discrete steps. The strategy prediction task in this case can be formulated as a sequence learning problem where we want to learn to predict the sequence of steps a student is likely to follow for a given problem. Knowing this sequence will provide the ITS prior information that it can use to adapt, e.g., by better tailoring the available hints and feedback. Sequence learning is applicable to many tasks in general with one of the most popular applications being machine translation where the goal is to translate a sequence of words from one language to another language. A widely used traditional approach that has been applied in sequence learning is Hidden Markov Models (HMMs). However, this assumes a one-step dependency where each step depends only upon the prior step. Student strategies in problem solving though are much more complex where a student action in one step can influence several downstream steps. Long Short-Term Memory (LSTMs) [11] are DNNs that can model such long-term dependencies in sequences. However, LSTMs are known to take an extremely long time to train for large datasets [39]. To address this, we propose a Neuro-symbolic model where we combine the semantics of a symbolic AI model called Markov Logic [9] with LSTMs. Markov Logic encodes domain-knowledge using first-order logic formulas. The formulas establish relationships in the data that can be described in the form of a graph structure. Our approach learns symmetries between instances based on the graph structure and then uses these symmetries to train an LSTM more efficiently. Specifically, we use *importance sampling* to choose a subset of training instances to make learning more efficient. While importance sampling-based learning in DNNs has been used previously to scale up training [1, 13, 14], most existing approaches determine the importance of a training instance by estimating the gradient norm which is computationally expensive [14]. In our case, we determine importance of a training data instance based on symmetries in the MLN graph. Specifically, the idea is that if several strategies are likely to be symmetrical then we can learn more efficiently from a smaller subset of strategies instead of the whole training set. To do this, we learn an embedding such that problem instances which have symmetries in the graph have similar vector representations in the embedding. We then cluster the embedding vectors and sample instances from each cluster to train the model. The idea is to sample a subset with same overall distribution of strategies as the original dataset. That is, we end up with a smaller dataset the preserves much if not all the information in the original dataset. However, since the clustering is approximate, we may end up with some clusters where the strategies are likely to be more diverse than others. Therefore, we adaptively train the model by updating importance weights for the clusters in iteration $t+1$ based on the trained model in iteration t. Specifically, we sample more data instances from a cluster in $t+1$ when the model trained in iteration t has smaller accuracy for instances sampled from that cluster.

We evaluate our approach on the publicly available KDD EDM challenge datasets [34]. We compare our approach with HMMs and pure LSTM methods that do not use symmetries in training the data and show that our proposed Neuro-symbolic model is more accurate and scalable, where we obtain high prediction accuracy by focusing on a small fraction of the training data.

2. RELATED WORK

Ritter et al. [30] provide a comprehensive survey on different approaches used to identify student strategies. Model tracing based tutors [4] have been previously used to identify strategies. In such cases, strategies may be pre-specified and the tutor can recognize correct and incorrect strategies. Model-tracing based methods have also been adapted to recognize new strategies [29]. Sequence learning has been widely used for strategy identification. Specifically, in Open Ended Learning Environments such as Betty's brain [23], student activities were captured in logs and sequence pattern mining methods was used on these logs to extract action sequences which in principle are similar to sequences that we consider in this paper. Different types of strategies based on these sequences were analyzed in multiple studies [16, 17, 18] which also mapped these sequences to performances to compare and analyze strategies followed by high performers to those followed by low performers. Sequence learning has also been used to extract strategies for self-regulated learning [3]. More recently, a study performed large-scale sequence pattern mining in MOOCs platform to analyze activity sequences of learners [37]. Further, in the context of conversational tutors, tutorial dialogues can be treated as sequence of actions based on language-as-action theory [2, 33]. These sequences which are akin to strategies are mapped into a taxonomy by education experts [26]. Approaches have been developed to recognize these sequences from natural language interactions to help automated tutors understand successful strategies to guide a student. In particular, sequence learning methods have been used for this task as well [32, 24]. Symbolic models such as Markov Logic have also been applied for recognizing these sequences using joint inference [36]. In general, Neuro-symbolic models have gained prominence recently and have found applications in problems that have graph structure. In [22], the authors provide a detailed survey of Neuro-symbolic models using graph neural networks. In complex problems such as visual question answering that require connections between language and image processing, Neuro-symbolic models have performed better than pure neural network based methods [38]. Our proposed application in this paper is further validation that Neuro-symbolic AI is a promising direction to solve complex problems.

3. SEQUENCE LEARNING MODELS

Student strategies can be defined in different ways. In particular, the definition of what constitutes a strategy also

depends upon the type of interaction the student has with the AIS. In our case, we only consider structured interactions with discrete steps. Therefore, we define the student strategy as a function of the sequence of steps in the interaction. Each step is characterized by the central concept that is utilized to solve that specific step, i.e., the knowledge component [20] (KC) used in that step. Therefore, we define strategy in our case as a sequence of KCs used by a student in a problem solving session. Note that, this formulation of strategy as a sequence of discrete components is similar to the definitions used in prior work [30]. Formally,

DEFINITION 1. *Given a student s and a problem p, we define the strategy as $\bar{\mathbf{x}}_{s,p} = K_{s,p}^{(1)} \ldots K_{s,p}^{(n)}$, where $K_{s,p}^{(i)}$ is the knowledge-component that s uses to solve the i-th discrete step in p.*

We can now formulate a learning problem as follows. Given training data, $\{\mathbf{x}_{s_i,p_j}\}_{i=1,j=1}^{n,m_i}$, where n is the number of students and m_i is the number of problems solved by the i-th student, we learn a model $\mathcal{P}$ such that for a student s' and problem p', $\mathcal{P}$ generates a sequence of knowledge components $K_{s',p'}^{(1)} \ldots K_{s',p'}^{(k)}$. Note that students sometimes use more than one KC per step, in this case, we just unroll these multiple KCs by repeating the step with each of the KCs. Therefore, for the rest of this paper, we treat both multiple KCs in a step and single KC steps without distinguishing them. Also, to keep notation simpler, instead of adding the subscripts s_i, p_j each time, we denote the training input and output pairs as $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$.

3.1 HMM Model

A popular model that is often used to learn from sequential data is the Hidden Markov Model (HMM), where we assume that every time-step is dependent on the previous time-step. HMMs are generative models represented using a dynamic Bayesian network. Each step in the time series is encoded by a *hidden-state* variable in the Bayesian network. We connect the hidden node corresponding to time-step j to the hidden node $j+1$ in the network and the observed feature at step j is also connected to the hidden node at step j. In our case, we encode the HMM as follows. Let O_t be the random variable representing the knowledge component at step t. We encode the knowledge component at step t as a vector $\mathbf{O}_t$ (the value of the random variable O_t) using a one-hot encoding. Let Z_t be the hidden-state variable corresponding to step t. The emission probability is given by $P(\mathbf{O}_t|Z_t)$ and the transition probability is given by $P(Z_t|Z_{t-1})$, i.e., the hidden state at time t depends only upon the hidden state at time $t-1$. The transition matrix is a $k \times k$ matrix that specifies transition probabilities to a state at time t given any other state at time $t-1$. Here, k which specifies the number of hidden states is pre-defined in the model.

The learning task in the HMM is to estimate the parameters of the HMM, namely, the transition matrix and the parameters of the emission probability distributions. Note that we need to estimate conditional probability conditioned on each possible state of the latent variable. To do this, we assume a Gaussian distribution represents the probability of each hidden state and the emission probabilities are also Gaussian distributions. Using the EM (Expectation Maximization) algorithm, we compute the parameters of the distributions using Max-likelihood estimation which has guarantees on convergence to a local optima. For predicting the strategy, we sample a KC at the first time step. Then, given a KC at any time step t, we generate the KC at time step $t+1$ as follows. Based on the observation, i.e., KC at step t, we predict an equivalent hidden state representation at step t and using the transition probability matrix, we sample the hidden state at time step $t+1$. We then predict the KC at time step $t+1$ using the emission probability.

3.2 LSTM Model

One of the problems with HMMs is that they have restrictive assumptions, i.e., each step depends only on the previous step. Ideally we would like to consider the student's activity across several steps to determine what his/her next step in the strategy is likely to be. For instance, suppose we have a student who works out a problem using a *divide-and-conquer* strategy, then there may be several small subproblems that the student solves before combining them together. In this case, a HMM model that simply looks at the previous step performed by a student may be able to capture the local strategy but will typically be unable to infer the global strategy since the dependencies may run across several steps. Therefore, to infer such advanced strategies, we need a more sophisticated model that captures longer-term dependencies. Long Short Term Memory (LSTMs) [11] are a variant of recurrent neural networks that have been used successfully for several problems like modeling text data. In particular, LSTMs can exploit longer range dependencies across words/sentences to learn a latent representation of sentences/documents. In our case, we apply LSTMs to learn a latent representation of the strategy.

Unlike HMMs, LSTMs are discriminative models that predict an output for step t based on the features observed in step t as well as a hidden state vector that summarizes the information up to step $t-1$. Note that a bi-directional LSTM can also consider information in steps succeeding t. To learn an LSTM for our task, we construct a tensor $T \in \mathbb{R}^{m \times n \times k}$, where m is the number of training instances, n is the number of steps and k is the dimensionality of features representing each training instance (s, p). Note that we can represent variable-length strategies using a special Start and Stop symbol in the LSTM to denote the start and end of strategies. The output of the LSTM for the t-th step is a vector representing the KC at step t. The final hidden state vector summarizes information for the full strategy for an input instance.

4. NEURO-SYMBOLIC MODEL

Though an LSTM can be directly used to learn a model for strategy prediction, it has certain limitations. LSTMs are known to converge very slowly for large datasets [39]. Further, LSTMs treat each instance in the training data as i.i.d (independent and identically distributed) which is limiting when there are underlying relationships among the instances. For example, problems are related to each other if solved by the same student, KCs used by the same student in similar problems are related to each other, etc. We address these limitations combining LSTMs with a symbolic model.

Neuro-symbolic AI [7], namely, combining symbolic AI mod-

els with DNNs has gained significant attention in tasks such as Visual Question Answering [38]. Neuro-symbolic models augment deep learning with knowledge from symbolic models. This can help learn deep models more efficiently even with limited labeled data [35]. Further, augmenting DNNs with symbolic models also controls overfitting. Specifically, since DNNs are highly expressive models, they are known to sometimes overfit the training data, particularly when the training data is non-diverse and in many problems such as image classification, data augmentation methods [5] seek to increase data diversity. In our Neuro-symbolic approach, we represent relationships in our dataset using the language of Markov Logic [9]. Based on symmetries in the relationships, we sample a smaller, more diverse training dataset to train the LSTM efficiently.

4.1 Markov Logic

Markov Logic [9] is a symbolic AI model designed as a representation and reasoning language for relational data. Markov Logic specifies relationships using the language of first-order logic. Each formula in Markov logic specifies a logical relationship using variables which can be substituted by symbols (also called constants or objects) from the data.

For example, we can model the fact that if two problems are similar then they require the same KC as a formula such as $\texttt{KC}(p_1, k) \wedge \texttt{Similar}(p_1, p_2) \Rightarrow \texttt{KC}(p_2, k)$. We can substitute this general formula using symbols in the data, say two problems P_1 and P_2 and KC K to obtain the formula, $\texttt{KC}(P_1, K) \wedge \texttt{Similar}(P_1, P_2) \Rightarrow \texttt{KC}(P_2, K)$. The formula that is substituted with the symbols is called a *ground-formula* using terminology from first-order logic. The predicates that are substituted with symbols are called *ground-atoms*, e.g., $\texttt{KC}(P_1, K)$ is a ground atom of predicate KC.

A Markov Logic Network (MLN) can be represented as a graph where the relationship encoded in each ground formula is represented by a clique in the graph. A clique is *activated* if the logical relationship specified by the ground-formula corresponding to that clique is evaluated to True in the data. For instance, in the above example, the clique corresponding to KC (P_1, K), Similar (P_1, P_2) and KC (P_2, K) is activated if the data asserts the logical relation that problems P_1 and P_2 use KC K and problem P_1 is similar to P_2. In MLN semantics, each activated clique represents a function parameterized by a weight attached to the ground formula. The full graph represents an undirected probabilistic graphical model [21]. For large datasets in real-world applications such as ours, the number of ground-formulas become very large resulting in an extremely large graph. In the standard use of MLNs, we learn parameters for the MLN based on Max-Likelihood Estimation (MLE). However, computing the gradient for MLE is infeasible when the graph is constructed from large datasets such as ours. Note that though, parameters for the MLN are required only if we want to use the MLN directly for probabilistic inference, i.e., when we want to answer queries using the MLN. While this is certainly desirable, it is well-known that MLN inference/learning algorithms cannot scale up to large datasets and perform extremely poorly in such cases [15]. Therefore, in our case, we make a simplifying assumption in the MLN that all the parameters for the graphical model have uniform weights. Thus, in our Neuro-symbolic model, Markov Logic is only used as a *language for knowledge representation (KR)* and not for inference/learning. That is, formulas specify relationships/connections in the MLN graph between different entities in our dataset (e.g. problems, students, etc.) while the actual learning and predictions are performed by an LSTM model. Note that in theory, other forms of KR such as Bayesian networks, arithmetic circuits or probabilistic programs can be used. However, the benefit of MLNs is that they specify relationships over large data using compact first-order formulas. Next, we describe the formulas in our MLN followed by how we learn symmetries in the graph to train the LSTM more efficiently.

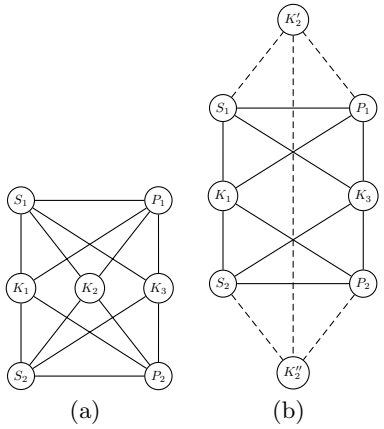


Figure 1: Illustrating symmetries in the MLN object graph. The graph represents symbols/objects in the MLN and the edges represent connection between variables, i.e., if objects appear together in a formula, they are connected in the graph. In (a) student S_1 can be exchanged with student S_2 and problem P_1 with P_2 to get an isomorphic graph. In (b) if the KCs K_1 and K_2 are similar, then the exchange is approximate.

4.1.1 MLN Structure

Our first set of MLN formulas relates the KCs to the problem and the student solving the problem.

$$\texttt{Student}(s) \wedge \texttt{Problem}(p) \wedge \texttt{PHierarchy}(p, h) \Rightarrow \texttt{KC}(s, p, t, k)$$

where s is a variable that denotes a student, p denotes a problem, t is a step, k denotes a knowledge component and h denotes the problem hierarchy which is the hierarchy of curriculum levels containing the problem, and PHierarchy(p,h) relates to the problem p in the hierarchy h where the hierarchy contains the curriculum unit name and the section name that the problem belongs to (e.g. Unit LCM, Section LCM-2).

Next, we encode the *homophily property* where the same KC is likely to be reused by a student for problems that are related to each other through a common problem hierarchy.

$$\begin{aligned}\texttt{Student}(s) \wedge \texttt{Problem}(p_1) \wedge \texttt{Problem}(p_2) \wedge \\ \texttt{PHierarchy}(p_1, h) \wedge \texttt{PHierarchy}(p_2, h) \wedge \texttt{KC}(s, p_1, t_1, k) \\ \Rightarrow \texttt{KC}(s, p_2, t_2, k)\end{aligned}$$

Next, we encode transitive dependencies between KCs by relating KCs that occur close to each other. Specifically, this encodes local dependencies across KCs (similar to a HMM model).

$$\texttt{KC}(s, p, t-1, k_1) \wedge \texttt{KC}(s, p, t, k_2) \Rightarrow \texttt{KC}(s, p, t+1, k_3)$$

4.2 Embeddings

Given the formulas, we can define the MLN object graph by connecting objects that appear in the same ground formula. For instance, consider an example MLN object graph shown in Fig. 1 (a) with two students (S_1, S_2), two problems (P_1, P_2) and three knowledge components (K_1, K_2, K_3). The edges indicate that in the graph corresponding to the MLN, there is a connection that relates the corresponding symbols. Note that S_1 works on P_1 and S_2 works on P_2, where P_1 and P_2 are related since they correspond to the same topic. Suppose both $S1$ and S_2 use the same strategy, K_1, K_2, K_3, then, we can exchange S_1 with S_2 and P_1 with P_2 to get an isomorphic graph structure. Now, suppose $S2$ uses a strategy K_1, K_2', K_3 that is slightly different from the strategy of S_1, say, K_1, K_2'', K_3, then we obtain a graph structure shown in Fig. 1 (b), where the new connections are shown by dotted lines. Now, exchanging S_1 with S_2 and P_1 with P_2 will not give us a graph structure that is isomorphic to the original graph. However, suppose that the knowledge components K_2' and K_2'' are similar to each other, i.e., there are many other problem instances where students use the KCs K_2' and K_2'' interchangeably, then exchanging S_1, P_1 with S_2, P_2 will yield an approximation that is still quite similar to the original graph. This means that using (S_1, P_1), it is reasonable to obtain a model that can predict the strategy for (S_2, P_2) and vice-versa. The set $\{(S_1, P_1), (S_2, P_2)\}$ is therefore an equivalence class consisting of approximately symmetrical instances. In general, if we group together approximately symmetrical instances in our training data, then we can train our LSTM model with diverse instances by sampling these groups since each group represents data that is likely to have a similar effect in training the model. We do this by learning *embeddings* for nodes in the graph structure.

Unfortunately, the size of the graph becomes very large as we increase dataset size and it is practically infeasible to construct the graph structure explicitly. Though several approaches have been developed that identify symmetries in MLNs using graph automorphism groups [27] using tools such as saucy [6], these approaches generally work directly on the graph structure. Further, it is possible to infer symmetries in graphs using other neural-network based methods such as Node2Vec [10] and Graph Convolutional Networks (GCNs) [19]. However, all these approaches work on general graphs and considering an MLN graph as a general graph is problematic since the graph becomes very large even for smallish datasets. This makes it hard to apply such approaches to our strategy identification problem since we expect to have an extremely large graph. Therefore, we instead use a recent, much more scalable Markov Logic graph specific approach called Obj2vec [12] that detects symmetries without explicitly constructing the graph. This approach is based on identifying approximate symmetries based on neighborhoods of a node using a neural network without constructing the actual graph.

4.2.1 Obj2Vec

Obj2Vec is inspired by skip-gram models [25] which are widely used to learn word embeddings. In skip-gram models, we learn an embedding for a word based on its *context*, i.e., the neighboring words that it typically appears with in text documents. For words which have similar contexts, we learn similar vector representations. Word2vec [25] is arguably the most popular skip-gram model, where we train a neural network for learning the embedding. Specifically, for each word w as input, the neural network learns to predict the context of w. Typically, The inputs and context words are encoded as one-hot vectors. The hidden layer in the neural network typically has a much smaller number of dimensions as compared to the input/output layers. The hidden-layer learns a dense, low-dimensional embedding, where similar words have similar vector representations. This is because, words that are similar typically have similar contexts in text documents and therefore the neural network learns a similar representation in the hidden layer for such words.

Obj2Vec extends the idea of word embeddings to MLNs. Specifically, recall that each ground formula of an MLN represents a clique in the MLN graph. For each activated ground formula, i.e., formula that represents a relationship that is supported by the data, we predict a symbol/object in the formula from other symbols/objects in that same formula. For example, suppose our data shows that *Alice* and *Bob* use the knowledge component `Slope-Intercept` across several problems. Then, all ground formulas that contain either *Alice* or *Bob* and the KC `Slope-Intercept` are activated. In this case, both *Alice* and *Bob* are said to share a common context. Therefore, we predict the symbol `Slope-Intercept` from both *Alice* and *Bob*. That is, we have an autoencoder neural network where the input is a one-hot encoding of *Alice* (or *Bob*) and the output is a one-hot encoding of `Slope-Intercept`. The neural network must therefore learn a common representation for both *Alice* and *Bob* since it needs to make similar predictions for both. Thus, suppose the hidden-vector representation (or embedding) for *Alice* is v_{Alice} and that for *Bob* is v_{Bob}, then $v_{Alice} \approx v_{Bob}$. Note that the embedding defines a continuous approximation of symmetries in relationships specified in the MLN graph. That is, the distance between the vectors v_{Alice} and v_{Bob} quantifies the symmetry between *Alice* and *Bob* based on relationships specified in the data.

4.3 Scalable Learning using Symmetries

The embedding vectors from Obj2Vec encodes relational knowledge from the MLN graph. Given the embedding vectors, we now train an LSTM to predict the student's strategy. Specifically, let our input instances be $\mathbf{x}_1 \ldots \mathbf{x}_N$, where each $\mathbf{x}_i$ consists of embeddings for a specific student s solving problem p, and the outputs are $\mathbf{y}_1 \ldots \mathbf{y}_N$, where $\mathbf{y}_i$ is a sequence of KCs used by the student s to solve the problem p. The LSTM training objective is given by,

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\psi(\mathbf{x}_i, \theta), \mathbf{y}_i) \quad (1)$$

where θ^* and θ represent the parameters of the LSTM, $\mathcal{L}$ is a loss function and $\psi(\mathbf{x}_i, \theta)$ is the sequence of KCs output

by the LSTM parameterized by θ for input $\mathbf{x}_i$. In general, a stochastic gradient descent (SGD) procedure can be used to minimize the objective in Eq. (1). In SGD, we sample the training instances to approximate the gradient. Typically, SGD assumes that all training instances are equally important, and therefore samples them uniformly. That is, the probability of sampling a specific instance in the training data is equal to $p = \frac{1}{N}$. However, this approach is expensive particularly if we repeatedly choose training instances that are similar to each other. For example, suppose all the training instances that we sample are likely to encode similar strategies, then our model may take a long time to understand diverse strategies. Further, since the underlying data encodes symmetries (from the MLN graph), the information in one training instance may be very similar to the information in another symmetrical instance. Therefore, we force the model to learn from instances with diverse relationships by imposing an importance distribution over the training data. Specifically, training instances with larger importance are more likely to be chosen as compared to training instances with smaller importance.

In general, to focus the training on more important data instances, we can modify the sampling distribution such that each instance is sampled with a non-uniform probability. This approach has been explored in prior work, where we scale up training by replacing the uniform distribution over the training instances with an importance distribution that quantifies how important a specific example is for the training process [14]. Previous work such as [14, 1, 13], have focused mainly on approximating importance as a function of the gradient norm which is hard to compute exactly. In [14], therefore, the authors propose an approximation to the gradient norm and use this to target important training examples. The focus in these approaches is to target the training examples that are likely to induce changes when updating the model parameters during backpropagation which can be shown to translate to a reduced variance in the gradient estimates. However, in our case, we have more information apriori in the embeddings to identify importance of a training example in terms of their relationships. Specifically, recall that the embeddings are based on symmetries in the MLN-graph which encodes relational knowledge. Thus, if two embeddings are similar, then it means that they share similar relationships. For example, if two student embeddings are similar, then it is likely that for the problems both students have solved, their strategies use similar KCs. Thus, using embedding-similarities, our model focuses the training effort on instances that encode diverse relationships.

4.3.1 Adaptive Importance Weighting

Given the instances $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$, we cluster the instances using K-Means clustering. Each instance internally has two components, the student embedding as well as the problem embedding. Since we want to exploit symmetries in both, we cluster them along both dimensions. Let $\{\mathbf{C}_s^{(i)}\}_{i=1}^{n_1}$ and $\{\mathbf{C}_p^{(i)}\}_{i=1}^{n_2}$ denote the clusters found by K-Means using the student embeddings and the problem embeddings respectively, where n_1 and n_2 are the number of clusters. We now sample from each cluster to obtain a reduced set of training examples. For each cluster, we assign an importance weight to quantify how often we need to sample that cluster. Let q_i represent the importance weight of the i-th

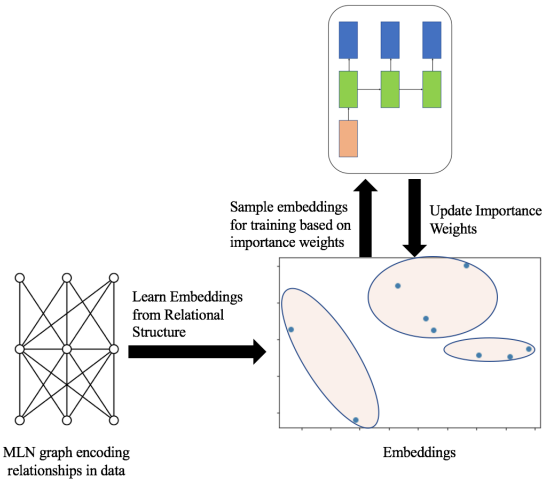


Figure 2: Schematic illustration of our proposed approach. We learn embeddings based on symmetries in the relational data. We then learn a one-to-many LSTM to map an input instance to a sequence that represents the strategy by focusing the training on a sample of the data that reflects symmetries in the data.

cluster. Ideally, we want to quantify q_i based on the symmetries encoded within the cluster. Specifically, if the cluster contains highly symmetric instances, then we require fewer training examples from the cluster. On the other hand, if the cluster contains diverse instances, we require more training samples from that cluster. One way to quantify this is using traditional clustering metrics such as within-cluster sum of squared errors (SSE) to measure cohesion within the cluster. That is, if the embedding vectors are close to each other within a cluster which implies a small SSE, then it is indicative that we may need fewer samples from that cluster. However, this approach may not necessarily yield the best results since both the embeddings and the clustering are simply approximations to true symmetries in strategies. For example, suppose the embedding vector for *Alice* is close to the vector for *Bob*, this means that considering all problems, *Alice* and *Bob* are approximately symmetrical. Similarly, if the vector for problem P_1 is close to problem P_2, this means considering all students, P_1 and P_2 are approximately similar. However, this may not always necessarily imply that the strategy followed by *Alice* for problem P_1 is guaranteed to be symmetrical to the strategy followed by *Bob* for P_2. Therefore, we use an adaptive approach to progressively learn these symmetries.

For training the model, from each cluster, we may require varying number of samples. That is, from clusters representing less symmetrical instances, we may require more samples while from other clusters that contain more symmetrical instances, we may require fewer samples. To account for this, we adapt the sampling as follows. We define the initial importance weight for the i-th cluster as $q_i^{(0)} = \frac{1}{K}$, where K is the number of clusters. Let $\theta^{(j)}$ be the parameters learned by the LSTM in iteration using samples from the clusters in iteration j. In iteration $j+1$, we update the importance

weight for the clusters as,

$$q_i^{(j+1)} = \frac{1}{m}\sum_{k=1}^{m}\mathcal{L}(\psi(\mathbf{x}_k^{(i)}, \theta^{(j)}), \mathbf{y}_k^{(i)}) \quad (2)$$

where $\mathbf{x}_k^{(i)}$ is a training example that consists of a randomly sampled instance from the i-th cluster. For example, if we are updating the i-th student cluster, $\mathbf{x}_k^{(i)}$ is a randomly sampled student s from this cluster combined with a problem p, where p is a problem that has been solved by s. Thus, if the LSTM in iteration j effectively encodes the symmetries in the i-th cluster, then the loss in Eq. (2) is likely to be small. Thus, $q_i^{(j+1)}$ is small. On the other hand, if $q_i^{(j+1)}$ is large, then we need more samples from the i-th cluster since the LSTM trained using the samples collected until iteration j does not effectively model the instances in the i-th cluster. The importance weights for the i-th cluster adaptively change from $q_i^{(0)}$ $q_i^{(1)}$ ... $q_i^{(T)}$. Note that each iteration adds training data to the LSTM and thus effectively increases training time. However, we do not begin the training from scratch in each iteration. Specifically, for iteration $j+1$, we consider the LSTM learned until iteration j as the starting point. This is similar to *pre-training* that is common in deep network training. For iteration $j+1$, the $\theta^{(j)}$ represents the pre-trained parameters of the LSTM. We stop adapting the weights after a fixed number of iterations depending based on a cutoff for the training time. Note that more advanced convergence criteria can be explored here which is a part of our future work. To summarize, Fig. 2 shows a schematic representation of our overall model.

5. EXPERIMENTS

5.1 Setup

We evaluate our approach on the publicly available KDD EDM challenge datasets, `Algebra 2008, 2009` and `Bridge to Algebra 2008, 2009` [34] which consists of data collected from the Mathia platform. Each instance consists of several discrete steps and each step is mapped to a knowledge component which is used to solve that step. The statistics for the two datasets are shown in Table 1. As shown in the table, these datasets are quite large with over 850K and 1.6M instances respectively. All our experiments were performed on a 64GB memory machine with a Nvidia GPU and an Intel Core-I9 processor. For computing accuracy, in each input instance, we compute the percentage of total steps where the true KC matches with the predicted KC. The overall accuracy is computed as the average accuracy across all instances. To measure variance of our estimates, for each of the results shown, we run the experiments 10 times and compute the mean accuracy and the standard deviation of the accuracy. Next, we describe the implementation of different approaches that we use in our experiments. The code and data for our implementations are available here[2].

5.1.1 Hidden Markov Model

We trained a Gaussian Hidden Markov Model (we refer to this as `HMM`) using the sklearn package in python. We set the number of hidden states to 100 and initialized the Gaussian emission probabilities with a full covariance matrix so that it has flexibility to generate varied sequences. We trained the model using the EM algorithm.

[2]https://github.com/anupshakya07/SSPM

Dataset	Total Instances	No. of Students	No. of Problems	No. of unique KCs
algebra_2008_2009	838728	3310	188368	541
bridge_to_algebra_2008_2009	1624951	6043	52754	933

Table 1: Details of the dataset.

Learning-rate	Optimizer	Batch-size	Dropout-rate	LSTM (hidden state)	Obj2Vec embedding
0.001	Adam with CCE-Loss	100	0.3	200 dimensions	300 dimensions

Table 2: Parameters for training.

5.1.2 LSTM and Neuro-symbolic Models

We implemented a one-to-many LSTM using TensorFlow and Keras. For the pure (or vanilla) LSTM, we encode inputs as one-hot-encoded vectors representing the studentID, problemHierarchy and problemName. For the Neuro-symbolic model, we vectorize each instance, using a publicly available implementation of Obj2Vec [12] using the MLN formulas as specified in the previous section. Obj2Vec internally uses Gensim [28] to compute the embeddings for each symbol in the MLN. We use the embedding vectors of studentID, problemHierarchy and problemName as input to the LSTM encoder. Special Start and End tokens are used in the decoder section of the LSTM to identify the start and end of a prediction. The decoder unit predicts the KC at each time step, until an End token is found. To train the models in a feasible manner, we used a timeout of 3 hours. Within this timeout period, it was infeasible to use the all the instances since the training for the LSTM did not converge. Therefore, we randomly sampled instances to train our model within the specified limit. We refer to the trained models using random sampling as `LSTM-Random` and `LSTM-NS-Random` for the vanilla LSTM and the Neuro-symbolic models respectively. We further implemented a stratified-sampling/group based training on students and problems. For sampling by student, we selected N students from the student pool and for each selected student, we sampled M problems solved by that student. For sampling by problems, we selected N problems from the problem pool and sampled M students who have solved those problems. By increasing values of M and N, we progressively increased the instances as we show later in the results section. We refer to this as `LSTM-NS-NaiveGroup`.

5.1.3 Adaptive Training

We implemented K-Means clustering to cluster the data based on the embeddings and sampled from these clusters. We implemented a non-adaptive training model as follows. We independently clustered the students and the problems to generate C_1 student clusters and C_2 problem clusters. We then sampled one student from each student cluster and one problem from each problem cluster nearest to the cluster centers to create a training set of at most $C_1 * C_2$ instances that effectively covers all instances in our training data. We increase the number of clusters progressively starting from 100 student clusters and 1000 problem clusters to increase the number of instances in training as we show later in our results. We refer to this approach as `LSTM-NS-Clustered`. For our proposed adaptive weighting approach, we sample each cluster according to an importance weight. In each iteration, we update the importance weight of a cluster based on predictions made on a randomly sampled set of instances that were not used in training from that cluster. Here, we used 100 student clusters and 1000 problem clusters and

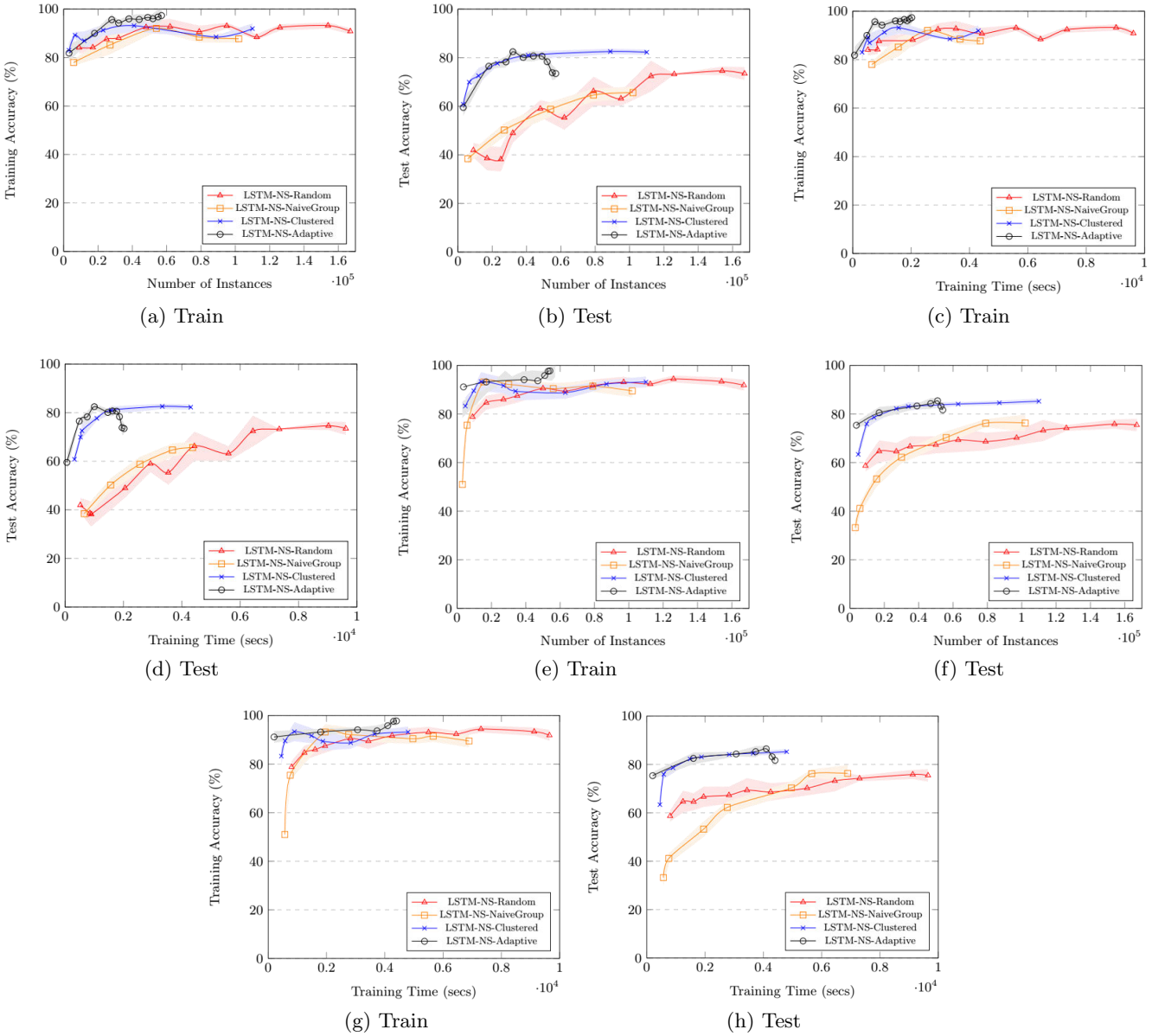


Figure 3: Accuracy results, the shaded portions show the standard deviation and the mean accuracy is plotted in the graphs. (a)-(d) corresponds to `Bridge to Algebra 2008, 2009` results and (e)-(h) corresponds to `Algebra 2008, 2009` results.

changed the importance weights of the clusters adaptively. We refer to this approach as `LSTM-NS-Adaptive`. The parameters for training our models are shown in Table 2.

5.2 Results

Fig. 3 compares the accuracy for different approaches. Fig. 3 (a), (b) shows the training and test accuracy respectively as we vary the number of instances used in training the models for the `Bridge to Algebra 2008, 2009` dataset. As we can see from these plots, `LSTM-NS-Adaptive` obtains the highest accuracy compared to the other methods. `HMM` (not shown in figures) gave us an accuracy of less than 5%. This low accuracy indicates that the strategy for diverse (or asymmetric) groups of students (or problems) cannot be represented by the same transition matrix. One possible approach that we will explore in future is to integrate our approach with HMMs, i.e., we learn an ensemble of HMMs where a HMM learns strategies for a symmetric group. A naive LSTM without embedding vectors, i.e., where inputs are a simple one-hot encoding of student and problems, also yields poor accuracy (less than 10%). This illustrates the importance of the latent features in the embedding vectors.

Note that we have only shown the results for the best performing model with latent dimension as 200 (experiments were carried out with different dimensions). As shown in Fig. 3 (a), (b), `LSTM-NS-Adaptive` and `LSTM-NS-Clustered` require a small fraction of the number of instances to obtain accuracy that is higher than `LSTM-NS-Random` and `LSTM-NS-NaiveGroup`. Further, the variance in accuracy of `LSTM-NS-Random` and `LSTM-NS-NaiveGroup` is much higher as shown by the shaded portion around the line plots compared to the variance of `LSTM-NS-Clustered` and `LSTM-NS-Adaptive`. `LSTM-NS-Adaptive` also obtains higher accuracy than `LSTM-NS-Clustered` as we adapt the weighting. However, note that the test accuracy starts to dip after the accuracy hits a peak value for `LSTM-NS-Adaptive`. This is because the adaptive cluster weights may focus too strongly on certain clusters in the training data which causes the LSTM to overfit. Therefore, in practice, we can stop the adaptation based on a validation set. Further, we can clearly see that exploiting symmetries in training leads to better generalization in Fig. 3 (b) where we see a significant difference between the accuracy for `LSTM-NS-Adaptive` and `LSTM-NS-Clustered` as compared to `LSTM-NS-NaiveGroup` and `LSTM-NS-Random` at smaller training sizes. Fig. 3 (c) and (d) also show the using relational symmetries in training results in a significant improvement in scalability since it shows that we can train `LSTM-NS-Adaptive` and `LSTM-NS-Clustered` in around half an hour to achieve an accuracy that is higher than the accuracy we obtain even after around 3 hours of training time in approaches where we do not choose training examples based on symmetries.

The results for the `Algebra 2008, 2009` dataset are similar to the ones for `Bridge to Algebra 2008, 2009`. As seen in Fig. 3 (e) and (f), the `LSTM-NS-Adaptive` model is the best performing model in terms of accuracy and it uses a small number of training instances to achieve this accuracy. Similar to the previous results, the variance for `LSTM-NS-NaiveGroup` and `LSTM-NS-Random` is much larger than that for `LSTM-NS-Adaptive` and `LSTM-NS-Clustered`. The training time shown in Fig. 3 (g) and (h) follow a similar pattern where `LSTM-NS-Adaptive` and `LSTM-NS-Clustered` can achieve high accuracy scores even with short training times since they take advantage of relational structure in the data.

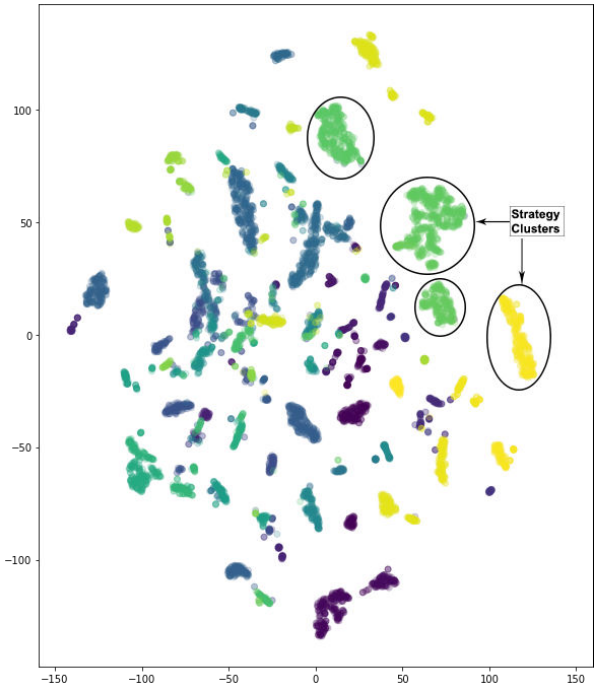


Figure 4: T-SNE visualization of strategies. The hidden layer of the final step in the LSTM is visualized for 100 test problems over all students. T-SNE reduces the latent LSTM vector to 2-D for visualization. Data points close together correspond to approximately similar strategies.

Table 3 shows the accuracy of predicting strategies for different problem units. Specifically, each problem in the dataset corresponds to a specific unit and we evaluate the models by testing the trained model on problems specific to a unit. For lack of space, we have not provided an exhaustive set of results for all units since there were around 50 units in the dataset. Instead, we provide accuracy results for the 10 units with largest number of data instances from the `Bridge to Algebra 2008, 2009` dataset. We see that on majority of the units, `LSTM-NS-Adaptive` has the best accuracy score. `LSTM-NS-Clustered` is the next best performing method. The difference in accuracy between units was significant in some cases. For instance, `LSTM-NS-Adaptive` had a very high accuracy for the unit `PERCENT CONVERSION` but a much lower accuracy for `ONE-STEP-EQUATIONS` and `TWO-STEP-EQUATIONS`. This may be due to higher complexity involved in solving equations as compared to problems involving percent conversion which may add to uncertainty in predicting strategies.

5.2.1 Structure in Strategies.

Finally, Fig. 4 shows a visualization of the strategies through a *T-SNE* plot. Specifically, we wanted to analyze if there are true patterns in the strategies. To do this, we use the `LSTM-NS-Adaptive` model to predict the strategy for 100 problems across all students. We show the results for `Bridge to Al-`

Unit	LSTM-NS-Random (%)	LSTM-NS-NaiveGroup (%)	LSTM-NS-Clustered (%)	LSTM-NS-Adaptive (%)
PROBABILITY	67.7	53.48	**77.45**	70.73
FRACTION-OPERATIONS-1	74.95	78.23	84.02	**90.49**
PERCENT-CONVERSION	99.01	99.18	88.6	**99.5**
SCI-NOTATION	66.67	69.39	69.9	**70.16**
PICTURE-ALGEBRA-2	90.23	**96.17**	93.29	95.17
RATIONAL-NUMBER-OPERATIONS	30.76	41.85	40.28	**66.13**
INTEGERS	87.47	78.83	93.52	**96.78**
ORDER-OF-OPERATIONS	60.03	50.35	45.53	**69.43**
ONE-STEP-EQUATIONS	**66.6**	53.36	58.86	58.06
TWO-STEP-EQUATIONS	**63.13**	63.03	59.42	61.5

Table 3: Comparing accuracy on test sets corresponding to different units in `Bridge to Algebra 2008, 2009`.

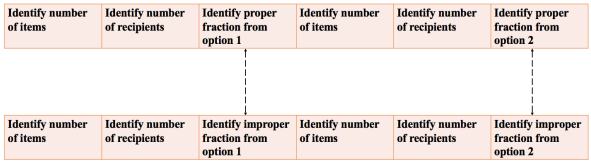


Figure 5: Strategies for different problems that have similar vector representations.

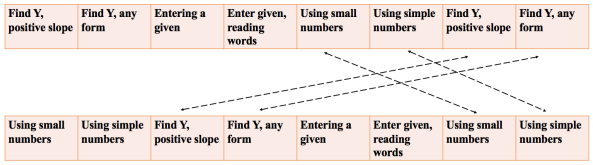


Figure 6: Strategies for the same problem that have similar vector representations.

`gebra 2008, 2009`. We use the hidden-vector from the last step of the LSTM as a representation of the strategy for a specific input instance. That is, this vector encodes information summarizing all the steps (or the full strategy) that the student performs when solving the input instance. We then plot this using the *T-SNE* plot that reduces the high-dimensional representation to a 2-D representation. Fig. 4 shows that there is a separation of different groups of strategies. The presence of such clusters of strategies indicates that there are indeed structures in student strategies and the representation learned by `LSTM-NS-Adaptive` discovers these symmetric structures.

Fig. 5 and 6 show two examples where the vectors representing the strategies are close to each other. Fig. 5 shows a case where the two strategies correspond to two different problems, one of which is related to proper fractions and the other to improper fractions. However, at a high level, these strategies are similar which is reflected in their similar vector representations. Another example shown in Fig. 6 shows a case where the two partial strategies shown correspond to the same problem are inversions of each other. That is, some of the steps in the two cases are performed in opposite orders. However, at a high-level, the strategies have symmetry which is reflected in their vector representations.

6. CONCLUSION

Predicting student strategies in problem solving can make AISs more engaging to students since the system can adapt itself to suit the student's strategy. In this paper, we described a Machine Learning approach to predict student strategies from large scale, structured student interaction data. Specifically, we adopted a Neuro-Symbolic approach, i.e., we combined LSTMs with a relational symbolic model to perform learning more efficiently. To do this, we encoded relationships in the data in the language of Markov Logic and based on relational symmetries in the data, we picked training instances are diverse. Doing this allowed us to learn our model to recognize diverse strategies at a smaller computational cost. Our evaluation on the KDD EDM challenge datasets show that our approach generalizes better and has significantly smaller training times as compared to approaches that do not exploit relational symmetries during learning. In future, we will extend our approach to datasets with finer-grained learner information and also develop joint inference models connecting mastery and strategies.

7. ACKNOWLEDGEMENTS

This research was sponsored by the National Science Foundation under the awards The Learner Data Institute (award #1934745) and NSF IIS award #2008812. The opinions, findings, and results are solely the authors' and do not reflect those of the funding agencies.

References

[1] G. Alain, A. Lamb, C. Sankar, A. C. Courville, and Y. Bengio. Variance reduction in SGD by distributed importance sampling. *CoRR*, abs/1511.06481, 2015.

[2] J. Austin. *How to Do Things with Words*. Oxford Press, 1962.

[3] M. Bannert, P. Reimann, and C. Sonnenberg. Process mining techniques for analysing patterns and strategies in students'self-regulated learning. *Metacognition and Learning*, 9(2):161–185, 2014.

[4] A. T. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In *Proceedings of the 8th International Conference on User Modeling 2001*, page 137–147, 2001.

[5] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, pages 113–123, 2019.

[6] P. T. Darga, M. H. Liffiton, K. A. Sakallah, and I. L. Markov. Exploiting structure in symmetry detection for cnf. In *Proceedings of the 41st Annual Design Automation Conference*, page 530–534, 2004.

[7] A. S. d'Avila Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *FLAP*, 6(4):611–632, 2019.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE Computer Society, 2009.

[9] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, San Rafael, CA, 2009.

[10] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 855–864, 2016.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[12] M. M. Islam, S. Sarkhel, and D. Venugopal. On lifted inference using neural embeddings. In *AAAI Conference on Artificial Intelligence*, pages 7916–7923, 2019.

[13] T. B. Johnson and C. Guestrin. Training deep models faster with robust, approximate importance sampling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 7276–7286, 2018.

[14] A. Katharopoulos and F. Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2525–2534, 2018.

[15] T. Khot, N. Balasubramanian, E. Gribkoff, A. Sabharwal, P. Clark, and O. Etzioni. Exploring markov logic networks for question answering. In *EMNLP*, pages 685–694, 2015.

[16] J. S. Kinnebrew and G. Biswas. Identifying learning behaviors by contextualizing differential sequence mining with action features and performance evolution. In *International Conference on Educational Data Mining*, pages 57–64, 2012.

[17] J. S. Kinnebrew, K. Loretz, and G. Biswas. A contextualized, differential sequence mining method to derive students' learning behavior patterns. *Journal of Educational Data Mining*, 5(1):190–219, 2013.

[18] J. S. Kinnebrew, J. R. Segedy, and G. Biswas. Integrating model-driven and data-driven techniques for analyzing learning behaviors in open-ended learning environments. *IEEE Trans. Learn. Technol.*, 10(2):140–153, 2017.

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[20] K. R. Koedinger, A. T. Corbett, and C. Perfetti. The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cogn. Sci.*, 36(5):757–798, 2012.

[21] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[22] L. C. Lamb, A. S. d'Avila Garcez, M. Gori, M. O. R. Prates, P. H. C. Avelar, and M. Y. Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. In *IJCAI*, pages 4877–4884, 2020.

[23] K. Leelawong and G. Biswas. Designing learning by teaching agents: The betty's brain system. *Int. J. Artif. Intell. Ed.*, 18(3):181–208, Aug. 2008.

[24] N. Maharjan, D. Gautam, and V. Rus. Assessing free student answers in tutorial dialogues using LSTM models. In *Artificial Intelligence in Education - 19th International Conference, AIED*, pages 193–198. Springer, 2018.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems*, pages 3111–3119, 2013.

[26] D. M. Morrison, B. Nye, V. Rus, S. Snyder, J. Boller, and K. B. Miller. Tutorial dialogue modes in a large corpus of online tutoring transcripts. In *Artificial Intelligence in Education - 17th International Conference*, volume 9112, pages 722–725. Springer, 2015.

[27] M. Niepert. Markov chains on orbits of permutation groups. In *UAI*, pages 624–633. AUAI Press, 2012.

[28] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, 2010.

[29] S. Ritter. Communication, cooperation and competition among multiple tutor agents. In *Artificial Intelligence in Education: Knowledge and media in learning systems*, pages 31–38, 1997.

[30] S. Ritter, R. Baker, V. Rus, and G. Biswas. Identifying strategies in student problem solving. *Design Recommendations for Intelligent Tutoring Systems*, 7:59–70, 2019.

[31] V. Rus, S. K. D'Mello, X. Hu, and A. C. Graesser. Recent advances in conversational intelligent tutoring systems. *AI Magazine*, 34(3):42–54, 2013.

[32] V. Rus, N. Maharjan, L. J. Tamang, M. Yudelson, S. R. Berman, S. E. Fancsali, and S. Ritter. An analysis of human tutors' actions in tutorial dialogues. In *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, FLAIRS*, pages 122–127, 2017.

[33] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[34] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Algebra I 2008-2009. Challenge data set from KDD Cup 2010 Educational Data Mining Challenge. Technical report, 2010.

[35] R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. In S. P. Singh and S. Markovitch, editors, *AAAI*, pages 2576–2582, 2017.

[36] D. Venugopal and V. Rus. Joint inference for mode identification in tutorial dialogues. In *COLING 2016, 26th International Conference on Computational Linguistics*, pages 2000–2011. ACL, 2016.

[37] J. Wong, M. Khalil, M. Baars, B. D. Koning, and F. Paas. Exploring sequences of learner activities in relation to self-regulated learning in a massive open online course. *Computers & Education*, 140:103595, 2019.

[38] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. B. Tenenbaum. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[39] Y. You, J. Hseu, C. Ying, J. Demmel, K. Keutzer, and C.-J. Hsieh. Large-batch training for lstm and beyond. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019.

Improving Automated Scoring of Student Open Responses in Mathematics

Sami Baral
Worcester Polytechnic Institute
sbaral@wpi.edu

Anthony F Botelho
Worcester Polytechnic Institute
abotelho@wpi.edu

John A Erickson
Worcester Polytechnic Institute
jaerickson@wpi.edu

Priyanka Benachamardi
Worcester Polytechnic Institute
pbenachamardi@wpi.edu

Neil T Heffernan
Worcester Polytechnic Institute
nth@wpi.edu

ABSTRACT

Open-ended questions in mathematics are commonly used by teachers to monitor and assess students' deeper conceptual understanding of content. Student answers to these types of questions often exhibit a combination of language, drawn diagrams and tables, and mathematical formulas and expressions that supply teachers with insight into the processes and strategies adopted by students in formulating their responses. While these student responses help to inform teachers on their students' progress and understanding, the amount of variation in these responses can make it difficult and time-consuming for teachers to manually read, assess, and provide feedback to student work. For this reason, there has been a growing body of research in developing AI-powered tools to support teachers in this task. This work seeks to build upon this prior research by introducing a model that is designed to help automate the assessment of student responses to open-ended questions in mathematics through sentence-level semantic representations. We find that this model outperforms previously-published benchmarks across three different metrics. With this model, we conduct an error analysis to examine characteristics of student responses that may be considered to further improve the method.

Keywords

Open responses, Automated scoring, Natural Language Processing, Sentence-BERT, Mathematics

1. INTRODUCTION

In many K-12 mathematics classrooms, teachers have come to rely on the use of open-ended questions to assess their students' knowledge and understanding of assigned content. Unlike close-ended problems, where there is a single or finite-number of accepted answers (e.g. a multiple-choice question), open-ended questions allow students to justify and express their thinking processes through language; it is common that students may combine language, images, tables, or other mathematical expressions, equations, and terminologies to illustrate their knowledge and understanding of the material.

While the use of open-ended questions is not found only in mathematical contexts, aspects of this domain make it particularly difficult to develop teacher supports for these types of question. Within computer-based learning platforms, research across fields of study have led to the development of a multitude of teacher-augmentation tools [1] and methodologies that leverage machine learning techniques. Among these supports, automated methods have been developed and deployed to help teachers assess student essays and short answers in several domains [25, 2, 3, 15]. As was highlighted in [9], the arduous task of manually assessing and providing feedback to student open-ended work may explain the decline of open-ended questions assigned over the course of a school year (e.g. Figure 1 which shows the number of open response questions assigned within the ASSISTments learning platform, aggregated over the last 10 years). In addition to this decline, as was also reported in [9], very few student responses to open-ended questions are ever scored by the teacher, with even fewer ever receiving feedback. Figure 2 illustrates this, as well as the subsequent plot of these values from February through October of 2020, during COVID-19 induced remote learning.

There are several notable challenges in developing automated supports to help teachers assess student open-ended work. It is also the case that student responses to open-ended questions differ in the context of mathematical and non-mathematical domains. One such difference, for example, is that many non-mathematical domains such as history or language arts, student "open-ended" essays and short answers are often comprised of multiple sentences and paragraphs [21, 25, 5, 8], whereas in mathematics, responses are generally shorter (maybe one or two, often incomplete sentences) [14, 9] that combine language with mathematical symbols, expressions, or other visuals. Aside from these response-level characteristics, however, several other student-, problem-, and even teacher-level factors can make the development of

Sami Baral, Anthony F Botelho, John A Erickson, Priyanka Benachamardi and Neil T Heffernan "Improving Automated Scoring of Student Open Responses in Mathematics". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 130-138. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

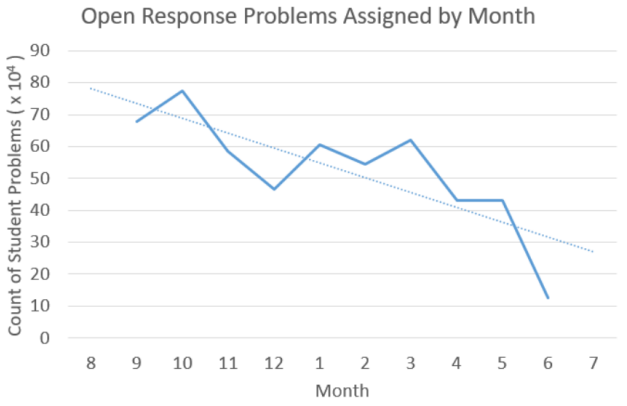


Figure 1: The number of open response problems assigned over the course of a school year with the ASSISTments learning platform, aggregated from 2010-2020.

these automated supports more challenging; consider, for example, the variation in how teachers approach the assessment of student answers, using different inherent rubrics and pedagogical philosophies [15, 17, 22, 23].

While the examination of student answers to open-ended poses challenges in developing automated assessment supports for teachers, prior work has shown promise in this context [9]. In that work, the authors explore several machine-learning and natural language processing (NLP) methods to predict teacher-provided scores to open-ended problems, offering an evaluation method and benchmark of comparison for similar methods[1]

In this paper, we build upon prior research presented in [9] to develop and evaluate an automated assessment model of student open responses in mathematics. We introduce a modeling approach using a sentence-level semantic representation of the student open responses to the existing models through Sentence-BERT (SBERT;[20]), using a novel reformulation of the "score prediction" problem. We compare our method to the previously-developed scoring models from [9], and subsequently apply an exploratory error analysis to identify areas of improvement that may be addressed by future iterations of these methods. Toward this, we seek to address the following research questions:

1. How does a model utilizing Sentence-BERT compare to previously developed approaches in predicting teacher given assessment scores for student response to open-ended problems?

2. What are the characteristics of student answers that correlate with errors observed in our Sentence-BERT model?

3. Which of student-, problem-, or teacher-level characteristics most explain the variance of error observed when the model is applied in real learning environments?

[1]The data and evaluation code from [9] was used in this work with permission from the original authors and in compliance with IRB.

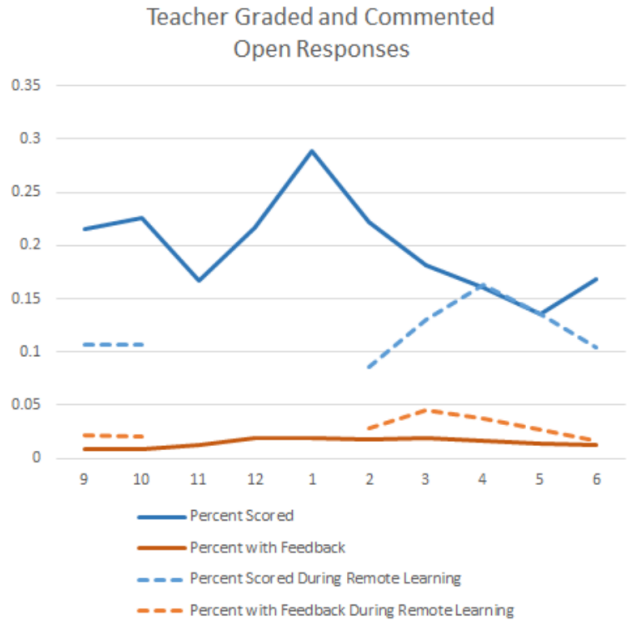


Figure 2: The percent of student open-response answers that were scored and given written feedback by a teacher before and during remote learning in response to COVID-19.

2. BACKGROUND

There have been several works related to the automated scoring of open-ended responses in the past. Most of such works utilize a combination of Natural language Processing (NLP) and machine learning techniques of ranging complexity to process open-ended responses. Much of the existing work in this area has been applied in the context of non-mathematical content. Developments such as C-rater[15] is a well-cited approach that uses such methodologies to estimate the assessed correctness of answers to short answer questions. This method uses grading rubrics and breaks down scores into multiple knowledge components to evaluate each student response. Other works [2, 3] have implemented clustering techniques to grade short textual answers to questions. More recently, studies have based their approach around deep learning methods, which have led to promising improvements over previous benchmarked results [21, 25]. While most of these works have been on non-mathematical domains, studies like [14] explore mathematical language processing using clustering techniques and the bag-of-words approaches for automated assessment of open-ended response in mathematics. However, this study only considers the mathematical content, discarding the non mathematical texts.

Many of these more-recent studies have utilized publicly-released embedding methods trained on large corpuses of data, including those of Word2Vec [18] and GloVe [19], to model the semantic meaning of words. However, word embeddings capture limited information about the semantics of a sentence, where the sequence of words may have large im-

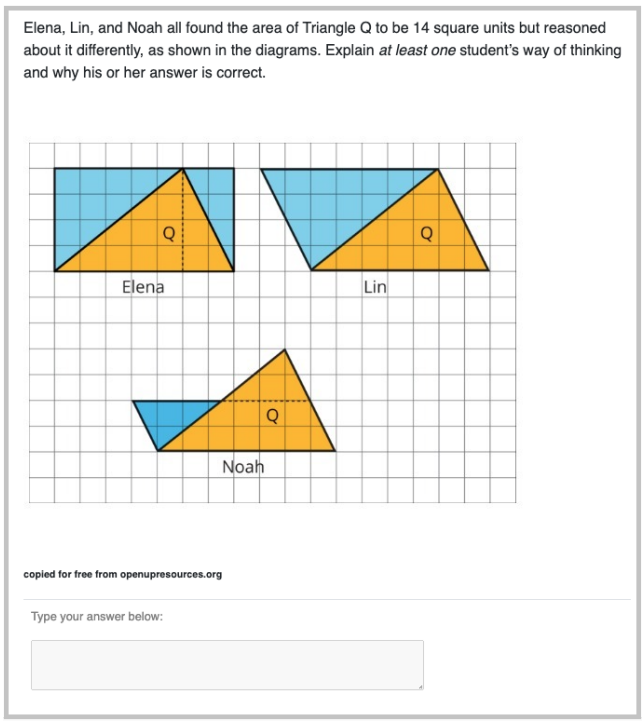


Figure 3: Example of an open ended question taken from openupresources.org

pacts on interpreted meaning. To capture the contextual information within sentences and further increase the generalization capabilities of NLP embedding methods, techniques such as Universal Sentence Encoders [4] and Sentence-BERT [20] generate a single embedding that is designed to be representative of the entire sentence while preserving the semantic and contextual information of the words within such sentences.

One of the most commonly-used NLP embedding methods in recent years has been that of Bidirectional Encoder Representations from Transformers (BERT, [7]). Building upon and distinguishing itself from other methods such as GloVe, the BERT method is designed to incorporate contextual information into generated embeddings to distinguish words that may have the same spelling but different meanings depending on usage (e.g. the word "bank" referring either a financial institution or perhaps a slope of land near a river); BERT has been shown to outperform many other approaches in a number of NLP tasks including, as is important for this work, semantic textual similarity (STS) [7]. Sentence-BERT, or SBERT [20], modifies the pre-trained BERT network to reduce the computational overhead of BERT in order to also generate a sentence-level embedding of a given series of words.

2.1 A Benchmark Comparison

In this work, we are exploring the use of this SBERT method to build upon the prior benchmark set in Erickson et al., 2020 ([9]) in assessing student answers to open-ended problems in mathematics. In that work, the authors discuss the challenges in developing models to predict teacher assigned grades for student open responses in mathematics, using a dataset of authentic student responses within the ASSISTments [11] learning platform. Erickson et al. compares 6 models utilizing machine learning (e.g. random forest and XGBoost [6]) and more complex deep learning (e.g. LSTMs [12]) techniques, combined with natural language processing algorithms to assess responses that are combinations of mathematical expressions and non-mathematical text. For the feature extraction process from the open response data, the study uses the Stanford Tokenizer [16] combined with Global Vectors for Word Representation (GloVe) [19].

3. METHODOLOGY

In this study, we build upon the work of [9] to develop and evaluate an automated scoring model based on the SBERT methodology; as will be detailed further, we refer to this model as the SBERT-Canberra model throughout the remainder of this work. Then, in a secondary analysis, we utilize real data collected from a pilot study of our model running within a computer-based tool that provides teachers with suggested scores to explore the limitations of our approach through an exploratory error analysis. Our data and approach to these analyses are described in this section.

3.1 Dataset

In this work, we utilize two datasets[2] of student answers to open-ended questions paired with teacher-provided assessment scores. An example of one of these open-ended mathematics questions is shown in Figure 3. In this example, students are not asked to find the area of the triangles, but rather explain in their own words what one of the figures is illustrating an approach to solving the problem.

For the development of our SBERT-Canberra model, we use the dataset (and evaluation code) from the Erickson et al. study [9]. This dataset is comprised of student answers to open response questions within the ASSISTments[11] online learning platform; the dataset consists of 150,477 total student responses from 27,199 unique students to 2,076 unique problems graded by 970 unique teachers. As was performed in [9], we omit any case where a student response contained no characters (e.g. an empty response or one containing only whitespace characters), or contained nothing but an image (cases where there was an image accompanied by other text or non-whitespace characters is not omitted). The removal of such empty responses resulted in the dataset dropping to 141,612 graded student responses, 25,069 unique students, 2,042 unique problems, and 891 unique teachers. Within this data, each response is accompanied by a teacher-provided assessment score that follows an integer ordinal 5-point scale from 0-4; a "4" here is synonymous with a student receiving a 100% for the response.

Table 1 lists several student answers contained within the dataset, chosen from across multiple problems for illustrative purposes. As was noted in the introduction, these responses highlight some of the challenges of this modeling

[2]The data and code used in this work cannot be publicly posted due to the potential existence of personally identifiable information contained within student open response answers. In support of open science, this may be sharable through an IRB approval process. Inqueries should be directed to the trailing author of this work.

Table 1: Sample student responses (selected from across multiple problems for illustrative purposes) and the teacher provided scores on a scale of 0 to 4 to the open ended questions in mathematics.

Sample Responses	Score
y=4x-2	4
I counted	4
I multiply -3 and 2x	2
diagram is on paper	3
Yes Because Y=mx+b	0
I got 2/9 by dividing by 4	3
I was not in class for this so I don't know.	1
I went multiplication first then division then multiplication	3
I got this by doing 45/75. I knew that 75 + 75 = 150 and 150 goes into 450 3 times and 3 x 2 = 6. So the answer is 6.	4
You would need an example and then you would need to draw a line and find out far away your shape is from the line and mark it and then do that on the rest of your lines on the shape	4
The distributive property means that a number outside a set of parentheses can be multiplied by each of the numbers within the parentheses and the answer will be the same. It works because it would be the same as multiplying each number by the number outside the parentheses and then adding them together.	1

task. First, the length of responses varies greatly between students as well as across problems. In addition to this, the interleaving of mathematics and linguistic text likely makes it difficult for pre-trained embedding models to interpret. Similarly, the variation in mathematical representation (i.e. the use of the term "dividing" rather than the "/" operator), may lead to confusion in a machine learning model trained over such data. As the mathematical variables are also represented by recognized english characters (e.g. "y"), it may be difficult to derive semantic meaning for such tokens. It is for this reason that we hypothesize that a contextual-based embedding approach, such as BERT and SBERT, may be superior to traditional embedding methods that do not account for context within the sentence. Finally, the noise in ground truth labels become evident from the table. The student who answered "I counted" but still received full credit, for example, exemplifies that some teachers may score students based on completion or other factors unrelated to their demonstration of understanding or mastery. This is not to say that any one scoring method is more correct or valid than another, but rather that there is likely large variation in these labels, making it difficult for machine learning models to effectively learn associations between student answers and these scores in some cases.

The second dataset used in this work is comprised of student responses collected during the pilot testing of a teacher-augmentation tool designed to aid in the assessment of student open response answers within ASSISTments [11]. This tool, called QUICK-Comments, used our developed model to predict the scores of student answers to open response questions in mathematics. Models were trained over the same open educational resource (OER) curricula from which the problems used in the first dataset were collected and produce estimates using the same grading scale as the first study. During the pilot study, 12 middle school mathematics teachers were given access to the tool and compensated for their time to assign, assess, and provide feedback to student open ended work during the Spring and Fall of 2020. This dataset consists of 30,371 graded student open responses to 915 unique open response problems solved by 1,628 unique students.

3.2 SBERT-Canberra Model

The model developed for this work follows a 2-stage process to generate estimates of teacher-assigned scores for a set of given student answers. In approaching this model, we propose a reframing of the initial problem. In [9], the problem was posed as a traditional supervised learning problem; in other words, given a set of student answers A, train a model $f(.)$ such that $Y = f(A)$. Instead, we propose a more unsupervised approach as depicted in Figure 4. If we have a set of historic answers $A_{0...n-1}$, and want to predict the score of a new answer A_n, a logical choice of score may be that corresponding with the historic answer that is most similar to the new answer A_n. In this way, the problem is posed as a similarity ranking problem rather than a supervised learning problem.

There are several potential advantages to this approach. First, when utilizing a pre-trained model of SBERT, described in Section 2, no actual model training is necessary (so long as a reasonable distance metric is identified). Second, as SBERT is optimized for contextual similarity tasks, the problem is better suited to utilize the embedding method's strengths. Finally, in a practical sense, as no model training is necessary (beyond utilizing the pre-trained embedding model), such a model can be more easily applied at scale, requiring just a pool of historic answers to compare against. We hypothesize that this method may also require fewer example answers than traditional machine learning methods as well, but this claim is not deeply explored in this current work.

In applying this method, the set of historic answers $A_{0...n-1}$ are fed through the pre-trained SBERT model to produce

Table 2: Features for the Linear Model of Error analysis of SBERT-Canberra model

Title	Description	Mean
Answer Length	Length of the answer	10.39
Average character per word	the average number of characters per words	3.54
Numbers count	total number of digits	3.54
Operators count	total mathematical symbols in the response	1.47
Equation percent	percentage of mathematical equations in answer	0.27
Presence of Images	Indicator of presence of images in the answer	0.15

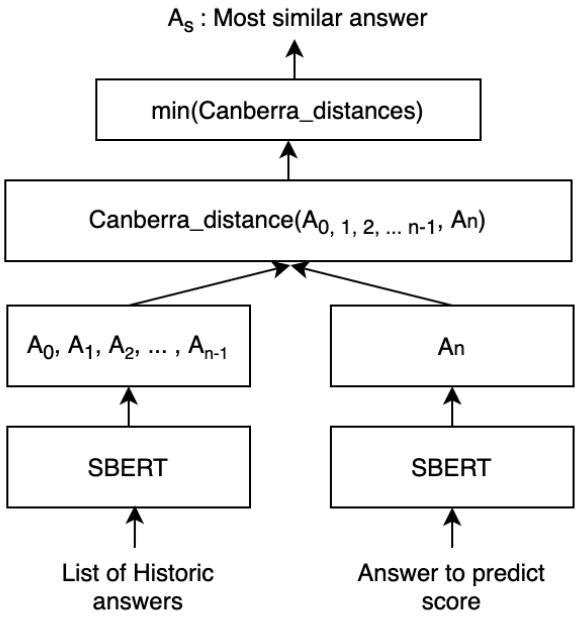


Figure 4: The design of the SBERT-Canberra method, that suggests scores based on similarity between the answers.

a 768-valued feature vector for each answer; these vectors are then stored for later access.. Given a new answer, A_n, a feature vector is similarly produced. In stage two of our method, all pairwise comparisons are then made between A_n and $A_{0...n-1}$, calculating Canberra distance [13] for each pair. Canberra distance, as opposed to other common distance metrics such as Euclidean or Cosine similarity, is a distance metric calculated over ranked lists. With this metric calculated for all pairs, the $A_{0...n-1}$ historic answers are then min-sorted to identify the most similar historic answer, A_s, to our new answer A_n. The score associated with A_s is then used as the prediction for the given answer A_n. The design to this approach is outlined in Figure 4.

As an additional component of this model, a "fallback" condition is implemented to be able to produce scoring estimates for problems where there are no historic answers on which to compare. In this case, we train a single multinomial regression model over all known answers, utilizing 1) the number of words in the answer and 2) the average length of each word in the answer; this model produces a probability distribution over 5 categorical labels (observing the 0-4 grading scale as a multinomial regression formulation). This one model is trained over all known answers and used then only in the case that no historic answers are available for the SBERT-Canberra model. This component is viewed as being part of our SBERT-Canberra approach.

3.3 Evaluation of SBERT

To evaluate our SBERT-Canberra scoring method, we utilize the same data and code presented in [9]. In that paper, the authors present the usage of a 2-parameter rasch model [24] (equivalent to a traditional item response theory, or IRT, model). The purpose of this model is to learn a separate parameter for each student and problem presented, representing student ability and problem difficulty, respectively. The intuition behind the use of this model is to evaluate an NLP automated scoring model based solely on its ability to interpret the words in each student answer. As the score of each answer is likely correlated with student ability (or knowledge) and problem difficulty (e.g. easy problems are likely to exhibit higher scores), such a model provides a reasonable minimum baseline of comparison. By adding a model's scoring estimates as covariates to the rasch model and then comparing the performance of such a model to the rasch model without covariates, we are able to observe the true value-added performance of the NLP scoring model.

Following the same procedure as conducted in [9], we are able to directly compare our Sentence-BERT method to those presented in that prior work. The models are trained and evaluated using a 10-fold student-level cross validation, and model performance is compared based on 3 performance metrics. First, treating the label as multinomial, rather than ordinal, AUC is caluclated using the method described in [10]. Second, the root mean squared error (RMSE), is calucalted over the ordinal prediction and label. Finally, a multi-class kappa is calculated, again using the multinomial label representation. The multinomial representations were argued to be appropriate due to the likely non-linear distribution of scores, while then RMSE provides insight into a more linear assumption of the data. Arguably an additional rank-based metric such as Spearman's Rho would also be a suitable metric of comparison, but is not included for more direct comparisons to the previous work.

3.4 Approach to Error Analysis of the SBERT-Canberra Method

In evaluating the SBERT-Canberra method, it is important to explore limitations of the approach in order to identify where the model does well and where it may yet improve through future iteration. As such, we also conduct an exploratory error analysis of the method using the data

Table 3: Rasch Model Performance compared to the models developed in Erickson et al.[9]

Model	AUC	RMSE	Kappa
Current Paper			
Rasch* + SBERT-Canberra	**0.856**	**0.577**	**0.476**
Erickson et al. 2020			
Baseline Rasch	0.827	0.709	0.370
Rasch + Number of Words	0.829	0.696	0.382
Rasch* + Random Forest	0.850	0.615	0.430
Rasch* + XGBoost	0.832	0.679	0.390
Rasch* + LSTM	0.841	0.637	0.415

*These rasch models also included the number of words.

collected from the QUICK-Comments pilot study. Toward this, we observe two regression models that observe absolute model error as a dependent variable. By exploring characteristics of student answers in the context of this modeling error, we can observe which aspects correlate most with higher prediction error. Similarly, we apply then a multi-level model to observe which of student-, problem-, and teacher-level identifiers most explains any observed modeling error.

3.4.1 Uni-level Linear Model

The uni-level linear model is based on student answer level characteristics. The student answer level characteristics are comprised of a set of six answer-level features extracted from the student open response data. These features are listed in Table 2. In calculating these features, the answer is first tokenized using the Stanford NLP tokenizer[16], dividing each textual answer into smaller tokens. For example, if the response to a particular problem is "I got 2/9 by dividing by 4", a simple tokenizer splits this response text by spaces which would give the list of tokens as: ("I", "got", "2/9", "by", "dividing", "by", "4"). Then from the tokenized data, we separate the tokens consisting of either digits or mathematical symbols. The number of such tokens is divided by the total number of tokens to calculate the equation percentage[3]. The average equation percentage calculated by the procedure mentioned above is 27% across the entire dataset. For calculating the length of the answer text, we count the total words in the text simply by splitting them by space. The average length of answers across the dataset is 10.39. Similarly, within each response, the number of numeric digits (i.e. Numbers count) and number of operator characters (i.e. Operators count) are counted independent of the tokens.

ASSISTments[11] allows students to upload images as part of the response to open-ended questions; this is most commonly a picture taken of work done on paper. The response text in such cases includes the URL of the uploaded image to the system. About 15% of the total responses in the dataset contains images. Some of such responses are entirely images, whereas in others, some text is provided as context. Since these scoring models are not yet designed to support images, we hypothesize that the images' presence contributes significantly to the modeling error.

A simple linear regression model is fit to the pilot study student answers, observing absolute model error as the dependent variable. This value is calculated by simply subtracting the predicted score from the teacher-provided label (as a linear label), and taking the absolute value. In this case, a value of 0 would indicate a correct estimate, while higher values (up to 4) represent greater prediction error; we do not differentiate between under- and over-predicting in this analysis.

3.4.2 Multi-level Linear Model

The uni-level linear model observes features that describe characteristics of the student responses, but as described in Section 3.1, modeling error may not be confined to just characteristics of the responses themselves. It is very likely that modeling error can be attributable to other external factors at the student-, problem-, and teacher-levels.

To explore this possibility, we apply a multi-level linear model observing the student answer characteristics as fixed effects, and student, problem, and teacher identifiers as three separate level-2 random effect variables. As it is the case that the same student may write multiple answers within our data, this structure is similar to that of a repeated-measure analysis.

$$\begin{aligned} abs(\text{model error}) = &\text{Answer Covariates} \\ &+ (1|\text{student identifier}) \\ &+ (1|\text{problem identifier}) \\ &+ (1|\text{teacher identifier}) \end{aligned} \quad (1)$$

Again observing absolute prediction error as the dependent variable, this analysis will be able to answer 1) whether the majority of explainable variance exists at the student-answer level or at a higher level, and 2) which of student-, problem-, and teacher-level identifiers most explains variance in our modeling error (e.g. which of these identifiers is most correlated with the error). The equation, expressed as its R code formulation, is reported as Equation 1.

[3]We acknowledge that this feature is a misnomer as it includes numeric terms, operators, and expressions as well as equations, but chose this feature name for sake of brevity.

Table 4: The resulting model coefficients for the uni-level linear regression model and random and fixed effects of the multi-level linear model of absolute error.

	Uni-level Linear		Multi-level Linear	
	Variance	Std. Dev.	Variance	Std. Dev.
Random Effects				
Student	—	—	0.034	0.185
Problem	—	—	0.313	0.559
Teacher	—	—	0.048	0.851
	B	Std. Error	B	Std. Error
Fixed Effects				
Intercept	0.581***	0.017	0.772***	0.070
Answer Length	-0.008***	0.001	-0.009***	0.001
Avg. Word Length	-0.014***	0.003	-0.013**	0.003
Numbers Count	<0.001	<0.001	<0.001	<0.001
Operators Count	-0.006***	0.001	0.002	0.001
Equation Percent	0.443***	0.018	0.080***	0.022
Presence of Images	2.248***	0.021	1.858***	0.028

*p <0.05 **p<0.01 ***p<0.001

4. RESULTS

4.1 SBERT Model

The results of the SBERT model is compared directly to the results from Erickson et al.[9] as shown in Table 3. As can be seen in that table, the SBERT-Canberra method outperformed the baseline as well as all previous models across all three metrics. While the difference in AUC values between our method and the previous best approach is notably small, the difference in both RMSE and Kappa appears to be comparatively larger. To interpret these two metrics, these results suggest that we should expect teachers to agree with our method's estimates 47% of the time accounting for random chance, and is likely to be wrong by just over half a grade-point on average. This also does suggest, however, that there is still plenty of room for improvement of these models.

What is also worth noting from the results of Erickson et al. [9], is the high performance of the baseline rasch model. This emphasizes the difficulty of this NLP modeling task in that the baseline model is using nothing other than the student and problem identifiers; it is able to seemingly predict teacher-provided scores with an AUC above 0.8 without using any part of the student response; there is only a 0.03 AUC difference between that baseline model and our current proposed method. This suggests that these external factors may be explaining a large portion of the student scores, and may subsequently explain a large portion of our prediction error.

4.2 Error Analysis of SBERT

In exploring this further, the results of the error analysis of the SBERT-Canberra method are presented in Table 4. It is found that the uni-level linear model explains 38.6% of the variance of the outcome as given by r-squared. Out of the six student answer-level features, nearly all were found to be statistically reliable predictors of model error; in verifying these results, it was found that all included covariates exhibited inter-correlations less than 0.3 (suggesting a moderately low impact of multicollinearity potentially skewing the interpretation of these results). In close examination of the coefficients of these features, however, despite being statistically reliable, many are found to be close to 0, suggesting a very little meaningful correlation with the modeling error. This is not the case, however, for two of these variables, Equation Percent and Presence of Images, we see a more meaningful coefficient. This suggests, due to the direction of this value, that the presence of mathematical elements as well as the presence of images (unsurprisingly) both correlate with higher prediction error. It further follows, then, that further improvements to the SBERT-Canberra method should explore methods of better representing and accounting for these mathematical terms in student responses; similarly, though likely much more difficult, incorporating an aspect of image recognition could be another area worth exploring.

In regard to the multi-level linear model, accounting for student, problem, and teacher identifiers each as random effects, we see that the inclusion of these level-2 factors explains some of the impact of the fixed effects (also in Table 4). Here it is found that all but two of the fixed effects are statistically reliable. It is also found that the magnitude of the coefficients for the Equation Percent and Presence of Images is also reduced. This suggests that, perhaps, the student and/or problem identifiers partially explain these correlations (some problems may be more likely to have responses with images or mathematical terms in them, or some students may be more inclined to use images or such terms more than others). What is worth noting, however, is that it was found that the level-2 variables account for 55.5% of the variance of the outcome. This suggests that a majority of the modeling error can be explained by these factors that are external to the student answers.

Looking at the variance of the random effects, it can be seen that the problem level identifiers contribute most in terms of explaining the variance of the outcome. It is certainly the

case that the SBERT-Canberra method is accounting for each individual problem in producing its estimates (e.g. it only observes historic answers within each unique problem), but it would seem that there are other problem-level factors that are not being accounted for within this approach.

5. LIMITATIONS AND FUTURE WORK

In regard to our approach as well as in light of our findings, there are several limitations and opportunities for future directions. While the SBERT-Canberra approach, utilizing sentence-level embeddings, outperforms the previously-developed models in predicting scores for open responses, the difference in AUC is rather small; the fact that the method produces a classification (as opposed to a probability as is often the case with such models) likely impacts its AUC performance. The manner in which the method makes its prediction can be considered a greedy approach in that only the closest historic answer is used to predict the score. Instead, a weighted vote approach using all historic scores (or a subset of similar scores above an identified threshold) may improve the model by allowing for some degree of uncertainty. Similarly, the use of the word count model as a fallback may further be improved; while it was the case that there were very few instances of problems not having enough data within the cross validation, improving this fallback method may help to improve the model when applied in practical settings where the "cold start" problem is more prevalent; as the method currently relies heavily on having a sufficiently-sized pool of human-scored historic answers, future research can focus on utilizing unlabeled student answers or exploring other unsupervised methods that may additionally support these methods in cases where labeled data is scarce.

While the SBERT-Canberra model performed arguably well, the error analysis revealed several areas where this approach, as well as others, may focus in future works. Most notably, as highlighted, the use of mathematical expressions and terms were found to be correlated with higher error; improving the representation of such elements can certainly be addressed in future work. A limitation of this, however, is that both models left variance unexplained in the outcome. We chose to look at these factors based on hypotheses and anecdotal observations, but there may be other large factors that can explain more of the error that we are seeing. Subsequent works could conduct more thorough surveys of both answer-level and higher-level factors. Future works can also explore additional model structures and language features that may lead to improvements to performance. The analyses presented in this work, however, can act as a baseline to further evaluate if future iterations of our approach truly improve upon these identified areas.

It is also the case that this work focuses only on models that predict numeric assessment scores, while we strongly believe that it will be equally, if not more important to additionally develop methods to suggest or generate directed feedback for for these student answers; teachers use textual feedback messages to offer constructive guidance to students, but it is often a very time-consuming task to write these messages for each students' answer. We believe that the SBERT-Canberra approach can be extended to support this task as well, where such a model may be able to recommend feedback to new student answers that has been previously given to an identified similar historic answer. Future work is intended to explore these methods further for such feedback-suggestion tasks.

6. CONCLUSION

In this paper, we have presented a novel approach in addressing and formulation of the problem of automating the assessment of student open-ended work. We have illustrated that our SBERT-Canberra method outperformed a previously-established benchmark, but still exhibits areas where it may be able to improve. Through the conducted error analysis, we have identified areas where more advanced methods of image processing and natural language processing (or math language processing), may lead to further improvements. With all of this, however, it was also identified that problem-level features appear to be most impactful in explaining the variance of modeling error; this is particularly surprising as variations in teacher grading were previously hypothesized to be a larger factor in this context.

With the findings from the study, our goal next is to use them to overcome the limitations mentioned above and guide our focus on improving the methods for assessment of open-ended questions in mathematics. It is the goal of this work to act as a step toward building better teacher supports for these types of open-ended problems, as well as provide others with guidance toward the same or similar goals.

7. ACKNOWLEDGMENTS

We thank multiple NSF grants (e.g., 1917808, 1931523, 1940236, 1917713, 1903304, 1822830, 1759229, 1724889, 1636782, 1535428, 1440753, 1316736, 1252297, 1109483, & DRL-1031398), as well as the US Department of Education for three different funding lines; the Institute for Education Sciences (e.g., IES R305A170137, R305A170243, R305A180401, R305A120125, R305A180401, & R305C100024), the Graduate Assistance in Areas of National Need program (e.g., P200A180088 & P200A150306), and the EIR. We also thank the Office of Naval Research (N00014-18-1-2768) and finally Schmidt Futures we well as a second anonymous philanthropy.

8. REFERENCES

[1] P. An, K. Holstein, B. d'Anjou, B. Eggen, and S. Bakker. The ta framework: Designing real-time teaching augmentation for k-12 classrooms. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2020.

[2] S. Basu, C. Jacobs, and L. Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics*, 1:391–402, 2013.

[3] M. Brooks, S. Basu, C. Jacobs, and L. Vanderwende. Divide and correct: Using clusters to grade short answers at scale. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 89–98, 2014.

[4] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Céspedes,

S. Yuan, C. Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[5] H. Chen and B. He. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1752, 2013.

[6] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] S. Dikli. An overview of automated scoring of essays. *The Journal of Technology, Learning and Assessment*, 5(1), 2006.

[9] J. A. Erickson, A. F. Botelho, S. McAteer, A. Varatharaj, and N. T. Heffernan. The automated grading of student open responses in mathematics. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 615–624, 2020.

[10] D. J. Hand and R. J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

[11] N. T. Heffernan and C. L. Heffernan. The ASSISTments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4):470–497, 2014.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] G. Jurman, S. Riccadonna, R. Visintainer, and C. Furlanello. Canberra distance on ranked lists. In *Proceedings of advances in ranking NIPS 09 workshop*, pages 22–27. Citeseer, 2009.

[14] A. S. Lan, D. Vats, A. E. Waters, and R. G. Baraniuk. Mathematical language processing: Automatic grading and feedback for open response mathematical questions. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*, pages 167–176, 2015.

[15] C. Leacock and M. Chodorow. C-rater: Automated scoring of short-answer questions. *Computers and the Humanities*, 37(4):389–405, 2003.

[16] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[17] S. L. Meier, B. S. Rich, and J. Cady. Teachers' use of rubrics to score non-traditional tasks: Factors related to discrepancies in scoring. *Assessment in Education*, 13(01):69–95, 2006.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[19] J. Pennington, R. Socher, and C. Manning. Global vectors for word representation. 2015.

[20] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[21] B. Riordan, A. Horbach, A. Cahill, T. Zesch, and C. Lee. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–168, 2017.

[22] J. Z. Sukkarieh, S. G. Pulman, and N. Raikes. Automarking: using computational linguistics to score short ,free- text responses. 2003.

[23] D. R. Thompson and S. L. Senk. Implementing the assessment standards for school mathematics: Using rubrics in high school mathematics courses. *The Mathematics Teacher*, 91(9):786–793, 1998.

[24] B. D. Wright. Solving measurement problems with the rasch model. *Journal of educational measurement*, pages 97–116, 1977.

[25] S. Zhao, Y. Zhang, X. Xiong, A. Botelho, and N. Heffernan. A memory-augmented neural model for automated grading. In *Proceedings of the fourth (2017) ACM conference on learning@ scale*, pages 189–192, 2017.

Topic Transitions in MOOCs: An Analysis Study

Fareedah ALSaad*
University of Illinois at Urbana-Champaign
alsaad2@illinois.edu

Thomas Reichel
University of Illinois at Urbana-Champaign
reichel3@illinois.edu

Yuchen Zeng
University of Illinois at Urbana-Champaign
yuchenz8@illinois.edu

Abdussalam Alawini
University of Illinois at Urbana-Champaign
alawini@illinois.edu

ABSTRACT

With the emergence of MOOCs, it becomes crucial to automate the process of a course design to accommodate the diverse learning demands of students. Modeling the relationships among educational topics is a fundamental first step for automating curriculum planning and course design. In this paper, we introduce *Topic Transition Map* (TTM), a general structure that models the content of MOOCs at the topic level. TTMs capture the various ways instructors organize topics in their courses by modeling the transitions between topics. We investigate and analyze four different methods that can be exploited to learn the Topic Transition Map: 1) Pairwise Constrained K-Means, 2) Mixture of Unigram Language Model, 3) Hidden Markov Mixture Model, and 4) Structural Topic Model. To evaluated the effectiveness of these methods, we qualitatively compare the topic transition maps generated by each model and investigate how the Topic Transition Map can be used in three sequencing tasks: 1) determining the correct sequence, 2) predicting the next lecture, and 3) predicting the sequence of lectures. Our evaluation revealed that PCK-Means has the highest performance in the first task, HMMULM outperforms other methods in task 2, while there is no winning in task 3.

Keywords

Topic Transition Map, Topic Transition, Word Distribution, Mixture Model, Hidden Markov Model, Clusters, Sequencing Tasks.

1. INTRODUCTION

For many decades, the process of creating courses has been a manual task that needs to be carefully managed by instructors and experts. However, with the recent advances in technologies and the emergence of Massive Open Online Courses (MOOCs), it becomes critical to automate the process of course design to accommodate the heterogeneity of online students and their diverse needs. According to [32], learning on demand is considered one factor that causes the high dropout rate in MOOCs. Learners have different learning demands depending on their motivations and goals. For instance, learners may seek knowledge about an interdisciplinary domain and hence need to learn modules from courses in several areas. This problem requires adopting a model in which MOOCs are used as modularized resources, rather than a set of pre-designed static courses. A crucial first step toward developing such a model is the automation of course plan design by sequencing lectures among different courses.

The main principle in designing the curriculum of any course is to organize course content according to some relations between topics. For instance, to help students to learn the materials, instructors carefully organize lectures as a sequence, based on the difficulty levels of topics [10, 27, 1] as well as the dependency relations between topics [11, 21, 23, 1]. The fundamental sequential structure of a course design is to place topics that are easy or prerequisite in earlier lectures while more advanced and dependent topics are taught in later lectures [1]. Consequently, modeling the relatedness among educational topics is a very crucial first step for automating curriculum planning and course design.

Modeling the content structure of MOOCs has recently attracted much research. Most of the current research has focused on modeling the prerequisite relationships between courses [29, 15], between lectures or segments of lectures [6, 7], or between concepts discussed within or across courses [3, 14, 17, 29, 15]. Using concepts to model MOOCs' content can be easily generalized to capture the relations in the concept space. However, because concepts are represented as keywords or phrases, it is hard to capture the different levels of granularity between lectures and courses. In addition, modeling prerequisite relationships between concepts cannot capture the various learning paths accommodated by different courses.

In this paper, we introduce the **Topic Transition Map**, a general structure that models the educational materials at the topic level. We model a course as a set of topics, and each topic is a set of concepts. Modeling content at the topic level is a more natural way to design custom course plans. We can think of a course as a path in the generalized Topic Transition Map. Thus, designing a new course becomes a task of identifying a path in the Topic Transition Map. Additionally, we investigate four methods that can be leveraged to construct the Topic Transition Map: *Pairwise Constrained K-Means* (PCK-Means) [2], *Mixture of Unigram Language Model* (MULM), *Hidden Markov Mixture Model* (HMMULM), and *Structural Topic*

*King AbdulAziz University, Jeddah, Saudi Arabia.

Fareedah Alsaad, Thomas Reichel, Yuchen Zeng and Abdussalam Alawini "Topic Transitions in MOOCs: An Analysis Study". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 139-149. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Model (strTM) [24]. We analyze and compare the Topic Transition Maps learned by these methods by studying how to exploit the Topic Transition Map in three sequencing tasks: 1) determining the correct sequence, 2) predicting the next lecture, and 3) predicting the sequence of lectures. To the best of our knowledge, we are the first work to introduce and investigate the use of Topic Transition Maps in modeling MOOCs content and sequencing lectures.

To evaluate the effectiveness of all methods, we use real MOOCs from three different domains: Python, Structural Query Language, and Machine Learning Clustering algorithms. Our evaluation revealed that while the PCK-Means has the highest performance on the task of finding the best sequence from a list of possible sequences, the HMMULM achieves the best performance on the task that predicts the next lecture in the sequence. Additionally, all methods perform similarly in the task of predicting the whole sequence with MULM has the lowest performance as it sometimes cannot predict the whole sequence. In addition to comparing various models in sequencing tasks, we visualize the Topic Transition Maps generated by different methods to qualitatively compare the resulted topic transition maps. We found that PCK-Means has extracted more meaningful topics with the best word distributions that clearly explain each topic.

The rest of the paper is organized as follows. In section 2, we present some of the related work. Section 3, defines the topics and topic transitions and states some applications of the topic transition maps. In section 4, we formally define our problem before describing the four different methods we exploit to construct the topic transition maps in section 5. Section 6 elaborates on our approach for the evaluation and the analysis of various models. Finally, we conclude our work in section 7.

2. RELATED WORK

Most of the work that models the content of MOOCs has focused on capturing the prerequisite relationships using different levels of granularity such as courses [29, 15], lectures or segments of lectures [6, 7], or concepts discussed within or across courses [3, 14, 17, 29, 15]. Modeling the relations between courses, lectures, or seqments of lectures is restricted to these units and cannot be generalized. While modeling dependency relations between concepts is considered a general structure that captures the required concepts before learning any concept, prerequisite relations cannot model the various learning paths accommodated by different courses. ALSaad and Alawini [2] have addressed this problem by proposing the precedence graph that captures the similarities and variations of learning paths among different courses. We build on their work and introduce the **Topic Transition Map** that maps each lecture to a topic and leverages the sequences of lectures among courses to capture the topic transitions pattern and hence the likelihood of such a transition. The main difference between the Topic Transition Map and the precedence graph is that Topic Transition Map models self transitions between a topic and itself and also captures how likely each topic to be the first topic in courses. While ALSaad and Alawini [2] have investigated the use of PCK-Means in modeling the precedence graph, in this paper, we explore three more methods in addition to PCK-Mean, namely MULM, HMMULM, and strTM, for modeling Topic Transition Maps. We also examine the impact of the learned topic transition maps on three different sequencing tasks. We believe that we are the first work that examines the use of topic transitions modeled from existing MOOCs to learn how to sequence new courses.

Some research has investigated the use of prerequisite relations between concepts to construct and sequence learning units [1, 16]. Both studies [1, 16] have developed supervised approaches based on feature engineering that extracted features from some external knowledge such as Wikipedia [1] and DBpedia [16] to infer the prerequisite relations between concepts. Our work is different as instead of modeling the prerequisite relations between concepts using supervised approaches, we model the Topic Transition Map or the various paths between topics using unsupervised methods, where a topic is a set of concepts. In addition, our methods rely only on the content of MOOCs without using any external knowledge. While Agrawal et al. [1] used the concept dependency graph to organize concepts to construct learning units and then sequence the learning units, we use lectures from existing MOOCs and investigate the impact of the learned transitions between topics to sequence lectures.

The most relevant research to our study is the work by Shen et al. [22]. Shen et al. [22] have proposed a method for linking similar courses to construct a map of lectures connected by two types of relations: similar and prerequisite. The constructed map only captures the similarity and prerequisite relations between certain units (lectures) and is not generalized to other lectures and thus cannot be used to predict the sequence of new lectures. In this paper, we map lectures to topics and construct the Topic Transition Map that depicts the precedence relations between topics and hence not tied with any specific units. Having a generalized Topic Transition Map can help in finding the sequence of lectures or predict the next lecture in the sequence as we discuss in section 6.2.

Another related line of research is the work on structural topic modeling by the Natural Language Processing, NLP, Community. In NLP, topic transitions have been used to model latent topical structures inside documents by assuming each sentence is generated from a topic where topics satisfy the first order Markov property [12, 25]. While Gruber et al. [12] only modeled the transition between topics as a binary relation (either remain on the current topic or shift to a new topic with a certain probability), Wang et al. [25] have developed a Structural Topic Model called strTM to explicitly model the topic transitions as probabilities that capture how likely one transits from a topic to another. Modeling transitions have been used in many applications related to NLP such as sentence ordering [25], topic segmentation [9], and multi-documents summarization [28]. In this paper, we investigate the use of topic transitions on modeling the topical structures in MOOCs by assuming a lectures is generated by one topic and use the sequences of lectures to learn the transitions between topics. We also explore the impact of using the Topic Transition Map to sequence lectures in three different sequencing tasks.

3. TOPIC TRANSITIONS

Before defining the topic transitions, it is important to briefly explain our representation of topics used in this paper. Similar to the definition of topics in the literature of the topic modeling research [5, 13], we define a topic as a distribution of concepts where concepts with higher probabilities tend to explain or characterize the topic. Concepts can be represented as words or phrases of words [3, 18, 26]. Each lecture is a composition of concepts and hence can be mapped to some topics. Depending on the length of lectures, lectures can cover one or more topics. Longer lectures usually cover more topics than shorter lectures. For example, traditional university lectures tend to be more elaborated and have longer duration than MOOCs lectures, which are usually concise and short in length. Therefore, the number of topics per lecture discussed in MOOCs is less than that of traditional university lectures. In this paper, since our work focus on learning the topic transitions from MOOCs, we assume

that each lecture is mapped only to one topic. This assumption is reasonable as lectures in MOOCs are concise and short in length. Having this assumption is also very useful as it helps in leveraging the sequences of lectures to learn the relations between topics.

A topic transition captures the precedence relations between topics. In other words, it means how likely instructors move or transit from one topic to another in the course delivery. It models the various ways of how instructors dynamically assemble concepts from the concepts space in order to construct the study plan of their courses. For instance, some instructors decide to start their *Python* course by explaining the topics: data types, conditional statements, loops, and then reading and writing from files. Other instructors may choose a different order, such as conditional statements, loops, string, and then lists. By leveraging the sequences of lectures from multiple courses, we can infer the latent topics of each lecture and hence model the common transition patters shared by multiple courses as well as the variations of different transitions or paths. To determine the strength or how common that transition is, each transition is attached with a score or a probability. For example, given the topics in Computer Science Programming: "Conditional Statements", "Loops", and "Arrays". It is more likely that instructors will explain the topic "Conditional Statements" immediately before the topic "Loops" and thus the topic transition score between them would be higher than the transition score between the topic "Conditional Statements" and the topic "Arrays".

Learning the topic transitions can be the initial block to facilitate several useful applications that can support modern learning. For instance, we can use the Topic Transition Map to extract the most common paths of topics in the field or explain the topic space in the current MOOC offerings. Learners can use transition maps to get more insights about the structure of topics in MOOC offerings. On the other hand, instructors can use these maps to improve their course offerings by examining the topic structure of related courses.

One important application of the Topic Transition Map is to support automatic curriculum planning and course design. Since courses consist of topics, learning the relations between topics would be the initial step to understand how likely instructors transit from one topic to another. We can think of a course as a path in the generalized Topic Transition Map. Thus, designing a new course becomes a task of identifying a path in the Topic Transition Map. In this paper, we analyze how can we use the learned topic transitions to sequence new courses.

4. PROBLEM FORMULATION

In this section, we formally formulate our problem. Given a set of courses $C = \{X_1, X_2, X_3, \ldots, X_N\}$ from a particular domain, where N is the total number of courses. We assume that courses in C are similar and hence have some content overlaps between them and also have the same difficulty level (e.g. Beginner, Medium, or Advance). A course X_i is represented as an ordered list of lectures $X_i = [x_{i1}, x_{i2}, \ldots, x_{i|X_i|}]$, where $|X_i|$ is the total number of lectures in the course X_i. Each lecture is a composition of concepts represented in some narrative way. In this paper, we assume a concept as a single word and hence lectures are represented using a bag-of-word representation.

Given the number of topics M, our goal is to map each lecture to a topic and leverage the sequences of lectures to learn topic transitions and construct the Topic Transition Map. The Topic Transition Map is represented as a matrix $\mathbf{A}$, where $\mathbf{A} \in \mathbb{R}^{M \times M}$. Each entry a_{ij} of the matrix $\mathbf{A}$ represents the likelihood of the transition from topic i to topic j. It reflect how common the precedence relation from topic i to j in the dataset courses. In addition to the Topic Transition Map, we also aim to learn the probability of each topic being an initial topic in courses. We denote the initial probability of each topic as a vector π, where $\pi \in R^M$. Along with the Topic Transition Map and the initial probability of each topic, it is important to model the word distribution of each topic, which represented as a matrix $\mathbf{B} \in \mathbb{R}^{M \times V}$, where V is the vocabulary size.

5. MODELING TOPIC TRANSITIONS

In this section, we explain the four different models we exploit to capture topics and Topic Transition Maps.

5.1 Pairwise Constrained K-Means

PCK-Means clustering algorithm [4] is a variation of the standard K-Means algorithm. To cluster instances, PCK-Means incorporates distance between points as well as pairwise constraints to guide the clustering process. Since the purpose of clustering is to capture topic transition patterns across courses, using PCK-Means helps to restrict the clustering process to cluster lectures across courses instead of within courses [2]. To guide the clustering, PCK-Means uses two types of constraints: **Must-Link** and **Cannot-Link**. Must-Link constraint determines lecture pairs that need to be clustered together, while Cannot-Link constraint specifies pairs that should not be grouped into the same cluster. To find the clusters, PCK-Means uses an objective function that minimizes both: 1) the distance between points (lectures) and the cluster centroid, and 2) the penalty costs of violating the constraints. For more information about PCK-Menas, please refer to [4].

Similar to ALSaad and Alawini [2], we use PCK-Means to build the Topic Transition Map $\mathbf{A}$. We first construct the list of Must-Link and Cannot-Link constraints to clusters lectures based on their content similarity into clusters. We assume that each cluster forms a topic and hence we need to learn the word distributions of each topic along with topic transitions. We link clusters by using the precedence relations between adjacent lectures and capture the strength of the transition by accumulating the frequency of transitions. To find the word distribution of each cluster or topic in the matrix $\mathbf{B}$, we accumulate the vector representations of each lecture that belongs to the same cluster. For more information, please see [2].

In order to estimate the initial probability π for each topic, we simply count the number of times of each topic being the first topic in the set of courses C. Then we do normalization to find the probability.

5.2 Mixture of Unigram Language Models

To capture topics, we use a mixture model of M unigram language models (MULM) with a bag-of-words representation. The mixture model is a generative probabilistic model that has been used for documents clustering. Thus, it will help in clustering lectures based on their topics, where each lecture belongs only to one cluster or one topic. In the mixture model, to generate a document, first one needs to choose the topic of the document according to the probability $P(\theta_i)$, where M is the number of topics, and then generate all the words in the document using the probability $P(w|\theta_i)$. According to the model, the likelihoods of a document x and the corpus C are calculated as follows:

$$P(x|\lambda) = \sum_{i=1}^{M} P(\theta_i) \prod_{w \in V} P(w|\theta_i)^{c(w,x)} \quad (1)$$

$$P(C|\lambda) = \prod_{j=1}^{N} P(x_j|\lambda) \tag{2}$$

To estimate the model parameters $\lambda = (\{\theta_i\}, P(\theta_i))$, where θ_i is the word distribution of topic i from the matrix $\mathbf{B}$, we use Expectation-Maximization algorithm [8] to find the parameters that maximize the likelihood of the data:

$$\bar{\lambda} = \arg\max_{\lambda} P(C|\lambda) \tag{3}$$

After learning the parameters, we map each lecture to a cluster or topic by maximizing the following equation:

$$c = \arg\max_{z_i} P(x|z_i) \tag{4}$$

Using the mixture model can help in clustering lectures according to their topics, however; it will not capture the transition patterns between topics or the initial probability of each topic. Therefore, similar to the PCK-Means method, we leverage the sequences of lectures to calculate the score of topic transitions and construct the Topic Transition Map $\mathbf{A}$. Likewise, we count the number of times courses start with each topic and normalize the results to model the initial probability π.

5.3 Hidden Markov Mixture Models

Instead of separately clustering lectures and then learning the transitions between them, using a Hidden Markov Model would allow us to jointly learn the word distributions of each topic ($\mathbf{B}$), the transition probabilities between topics ($\mathbf{A}$) as well as the initial probability of each topic (π).

The Hidden Markov Model, HMM, is a probabilistic graphical model that describes the process of generating a sequence of observable events according to some hidden factors [20]. It simulates how the real world sequence data is generated from hidden states. Particularly, it consists of two stochastic processes: 1) invisible process, and 2) visible process [30]. In HMM, invisible process consists of hidden states whereas visible process is observed sequence of symbols that are drawn from the probability distributions of the hidden states. Figure 1 demonstrates the HMM model. As you can see from Figure 1, each observable event in the sequence are generated from a hidden state and observations are conditionally independent given the hidden state. You can also notice that the hidden states form a Markov chain where each hidden state depends only on the previous state such as Z_{t+1} depends on Z_t.

To control the process of generating the observed sequences from hidden states, HMM has three parameters: π, $\mathbf{A}$, and $\mathbf{B}$. The first parameter, $\pi = \pi_1, \pi_2, \ldots, \pi_M$, is the initial probability distribution of each hidden state. The parameter π determines the probability of the Markov chain to start at each state and hence controls which state can be chosen as an initial state for the observed sequence. The second parameter, $\mathbf{A} \in \mathbb{R}^{M \times M}$, is the transition probability matrix that specifies how likely the model can transit from one state to another, denoted by $P(Z_{t+1}|Z_t)$ in Figure 1. The third parameter, $\mathbf{B} \in \mathbb{R}^{M \times V}$, is the emission probability matrix, where V is the total number of distinct symbols. It determines the likelihood of each state to produce each symbol, denoted by $P(X_{t+1}|Z_{t+1})$ in Figure 1. For example, to generate a sentence, a sequence of words would be drawn from the HMM model according to the three parameters π, $\mathbf{A}$, and $\mathbf{B}$.

In MOOCs, we only observe courses, where courses are sequence of lectures, while the topics of lectures and the transition between them are invisible or latent. Therefore, HMM would be a great model to simulate the generation process of courses and hence infer the latent states that contribute in the evolution of these lectures. In HMM, each hidden state generates only one symbol or word (see Figure 1). As our goal is to capture topic transitions using sequences of lectures as observed data, we map each lecture to a topic and assume each hidden state generates a lecture instead of a word. Our revised HMM assumes that each hidden state produces one lecture where each lecture is a bag-of-words. We ignore the sequence of words in lectures since the order of the words would not contribute to capturing the topic of each lecture. Figure 2 depicts the HMMULM utilized to capture the content of MOOCs.

In order to capture both the lectures' topics and the transitions between them, we combine the mixture model (MULM) with HMM, and we call the new model Hidden Markov Mixture of Unigram Language Model (HMMULM). To do that, we assume the Markovian assumption between topics where in the generation process, the choice of the next topic depends only on the current topic. Even though, the choice of the topic in the course delivery depends on the previous topics discussed so far, this simplified assumption makes sense due to the locality of reference property [1] of course design. Based on this property, when an instructor designs a course, a dependent lecture should appear as soon as possible after the prerequisite lecture to reduce students comprehension burden. Therefore, assuming the dependency between adjacent lectures not only simplifies the model but also aids in capturing the transitions between highly related topics. By combining the HMM with mixture model the likelihood of generating a course is as follow:

$$\begin{aligned}
P(X|\lambda) &= \sum_{all\ Z} P(Z|\lambda) P(X|Z,\lambda) \\
&= \sum_{all\ Z} P(z_1) P(x_1|z_1) \prod_{t=2}^{T} P(z_t|z_{t-1}) P(x_t|z_t) \\
&= \sum_{all\ Z} P(z_1) \prod_{w \in V} P(w|z_1)^{c(w,x_1)} \\
&\prod_{t=2}^{T} P(z_t|z_{t-1}) \prod_{w \in V} P(w|z_t)^{c(w,x_t)} \\
&= \sum_{i=1}^{M} \sum_{j=1}^{M} \pi(z_1 = s_i) \prod_{w \in V} B(z_1 = s_i, w)^{c(w,x_1)} \\
&\prod_{t=2}^{T} A(z_{t-1} = s_i, z_t = s_j) \prod_{w \in V} B(z_t = s_j, w)^{c(w,x_t)}
\end{aligned} \tag{5}$$

To estimate the HMMULM parameters $\lambda = (\pi, A, B)$, we use a modified version of Baum-Welch algorithm in order to model the observation sequences as a multidimensional categorical events. Following the work [19], we derived the equations of E-step and M-step to train the model and infer the transition probability between topics. In the E-step, we use the equations:

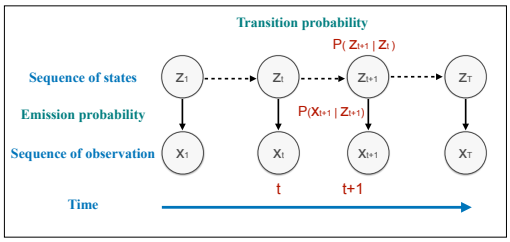


Figure 1: *The graphical model of HMM.*

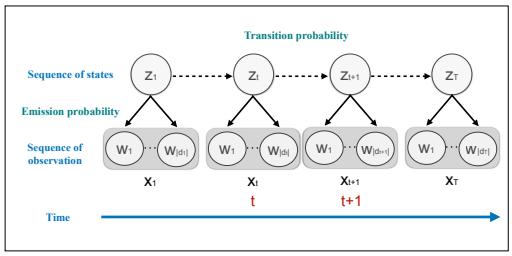


Figure 2: *The graphical model of HMMULM used to model the content of MOOCs.*

$$\gamma_t(i) = P(z_t = s_i|X, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{M}\alpha_t(j)\beta_t(j)} \quad (6)$$

$$\xi_t(i,j) = P(z_t = s_i, z_{t+1} = s_j|X, \lambda) = \frac{\alpha_t(i)A_{ij}\beta_{t+1}(j)\prod_{w\in V}B_j(w)^{c(w,x_{t+1})}}{\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_t(i)A_{ij}\beta_{t+1}(j)\prod_{w\in V}B_j(w)^{c(w,x_{t+1})}} \quad (7)$$

In the M-step, the following equations are used to choose the parameters that maximize the likelihood of the observed sequence of lectures:

$$\pi(i) = \gamma_1(i) \quad (8)$$

$$A_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \quad (9)$$

$$B_i(w) = \frac{\sum_{t=1}^{T}\gamma_t(i)c(w,x_t)}{\sum_{v=1}^{V}\sum_{t=1}^{T}\gamma_t(i)c(v,x_t)} \quad (10)$$

For more information about Baum-Welch algorithm, please see [20]. It is clear that the E-step and M-step equations are very similar to the standard HMM except that instead of emitting one symbol, HMMULM emits one lecture represented as a bag-of-words.

5.4 Structural Topic Model

The Structural Topic Model (StrTM) [25] is another probabilistic graphical model that functions very similar to HMMULM. It has been used to model the latent topical structures inside documents. Like HMMULM, it models topics and their transitions as hidden states that emit lectures as bags-of-words. Unlike HMMULM, strTM assumes each lecture as a mixture of content topics and functional topic. Functional topic, denoted by z_B, is used to filter out document-independent words that models the corpus background (or general terms) [31]. Each word in the lecture is either generated by one of the content topics or the functional topic:

$$w \sim \theta P(w|\beta, z_i) + (1-\theta)P(w|\beta, z_B) \quad (11)$$

where θ is the controlling parameter. According to strTM, the probability of lecture x_j being generated by some topic z_i is:

$$P(x_j|z_i) = \prod_{w\in V}[\theta P(w|\beta, z_i) + (1-\theta)P(w|\beta, z_B)]^{c(w,x_j)} \quad (12)$$

Another difference between strTM and HMMULM, is that strTM assumes the transition probabilities $\mathbf{A}$ and the emission probability $\mathbf{B}$ are drawn from Multinomial distributions and use the conjugate Dirichlet distribution to impose a prior on the Multinomial distributions:

$$\alpha_z \sim Dir(\eta) \quad (13)$$

$$\beta_z \sim Dir(\gamma) \quad (14)$$

Where η and γ are the concentration hyper parameters that control sparsity of α_z and β_z respectively.

To estimate the parameters of strTM, we use the expectation-maximization algorithm as described by [25]. For more information about strTM, please refer to [25].

Table 1: *The dataset utilized in the experiment.*

Domain	# of Courses	# of Lectures	Avg # of Lectures
Python	21	460	22
SQL	15	247	16
ML	10	99	10

6. EVALUATION

In this section, we first demonstrate our dataset and the parameters settings. Second, we compare different models by studying the impact of topic transitions learned from various models on three lecture sequencing tasks. Finally, we qualitatively evaluate the topics and their transitions.

6.1 Dataset and Parameters Settings

We collected our dataset from real online courses using various MOOC platforms and in three different domains: Python, Structural Query Language (SQL), and Machine Learning Clustering algorithms (ML). Table 1 presents the statistic of the dataset. We use 75% of the data as a training set and 25% as a test set. To choose the number of topics in each domain, we manually inspected the dataset to choose the number of topics. The number of topics for Python, SQL, and ML were set to 13, 10, and 9 respectively.

Each course in the dataset is represented as a sequence of lecture video transcripts. We preprocess lecture transcripts by eliminating stop words and some rare terms. After cleaning the data, we constructed the bag-of-word vector representations of all lectures. We only use lecture transcripts to represent lectures; therefore, we only need to set two thresholds (K_1 and K_2) of the PCK-Means method in order to select the list of Must-link and Cannot-link constraints. Since we do not have labeled data we chose the thresholds that maximize

Table 2: *The performance of **Task 1**: Finding the correct sequence using the permutation method. It is clear that PCK-Means achieves the highest performance.*

Dataset	Measures	Methods				
		Cosine	PCK-Means	MULM	HMMULM	strTM
Python	kendall's $\tau(\sigma)$	0.60	**0.73**	0.49	0.66	0.54
	Dir-P	0.37	**0.50**	0.43	0.28	0.31
	Undir-P	0.52	**0.56**	0.50	0.44	0.35
	Lev-Sim	0.60	**0.63**	0.45	0.49	0.50
SQL	kendall's $\tau(\sigma)$	0.58	**0.59**	0.58	0.55	0.41
	Dir-P	**0.46**	0.43	0.40	0.36	0.41
	Undir-P	**0.67**	0.58	0.49	0.44	0.41
	Lev-Sim	0.53	**0.57**	0.54	0.49	0.37
ML	kendall's $\tau(\sigma)$	0.68	**0.75**	0.63	0.58	0.64
	Dir-P	0.34	0.34	**0.44**	0.34	0.43
	Undir-P	0.52	0.52	**0.57**	0.41	0.53
	Lev-Sim	0.61	**0.65**	0.55	0.43	0.53

the *Silhouette Coefficient* clustering measure using the training data. We set $K_1 = 0.55$ and $K_2 = 0.004$ for Python, $K_1 = 0.8$ and $K_2 = 0.01$ for SQL, and $K_1 = 0.55$ and $K_2 = 0.01$ for ML. To set the hyper parameters of strTM method, we used a grid search and chose the values that maximize the likelihood of the training data. We set $\theta = 0.2$, $\gamma = 0.3$, and $\eta = 0.6$ for Python, $\theta = 0.1$, $\gamma = 0.3$, and $\eta = 0.1$ for SQL, and $\theta = 0.1$, $\gamma = 0.1$, and $\eta = 0.6$ for ML.

6.2 Sequencing Tasks

In this experiment, our goal is to compare the topic transitions modeled by different methods in three tasks: 1) Finding the correct sequence of lectures, 2) Predicting the next lecture given a sequence of lectures, and 3) Predicting the sequence of a list of lectures where the first lecture in the sequence is given. An example of real application for task 1 and task 3 is designing a new course plan by sequencing lectures before delivering them to students. However, task 1 and task 3 exploit two different techniques to find the sequence. In contrast, task 2 can be applied to recommend the next lecture to learners to customize their learning based on the history of lectures they already watched. In the evaluation, the purpose of each task is to compare different methods and evaluate the ability of the parameters (A, B, and π) of each model to find the correct sequence in the three different tasks.

6.2.1 Evaluation Measures

To compare different models, we use the sequences of lectures from courses in the test set as the ground truth sequences and exploit different measures to do the evaluation. First, we follow Wang et al. [25] and use kendall's $\tau(\sigma)$. Kendall's $\tau(\sigma)$ is an information retrieval measure that captures the correlation between two ranked list. It indicates how the predicted order differs from the ground truth where 1 means perfect match, -1 means total mismatch, and 0 indicates that the two orders are independent. Second, we use Levenshtein normalized similarity which is the opposite of Levenshtein normalized distance that measures the minimum number of edits (insertions, deletions or substitutions) required to transform the predicted sequence to the ground truth sequence. The goal is to find the sequence that has the Levenshtein normalized similarity close to 1 which indicates that the number of edits required is minimal. Third, we utilize the directed bigram precision (see equation 15) that captures the correctness of the order between adjacent lectures. The intuition behind using this measure is to evaluate whether the transition maps learned by different models have the ability to capture the correct direction order between topics and adjacent lectures. Finally, we use the undirected bigram precision shown in equation 16 to measure whether the transition map of each model can recognize adjacent lectures but incorrectly captured the direction between topics.

$$P_{Dir-bigram} = \frac{\#\ of\ correct(a \rightarrow b)\ in\ estimated\ sequence}{\#\ of\ correct(a \rightarrow b)\ in\ ground\ truth} \tag{15}$$

$$P_{Undir-bigram} = \frac{\#\ of\ correct\{a,b\}\ in\ estimated\ sequence}{\#\ of\ correct\{a,b\}\ in\ ground\ truth} \tag{16}$$

6.2.2 Task 1: Finding The Correct Sequence

To find the correct sequence of lectures, we follow the permutation method utilized by [25]. With courses that have large number of lectures, it is infeasible to find all the orderings of lectures. Therefore, when the number of the permutations exceeds 500, we randomly permutated 500 possible orderings of lectures as candidates. We ran the experiment 20 times for each method and recorded the average results.

In order to select the optimal sequence from the list of permutations in strTM and HMMULM, we follow Wang et al. [25] and choose the sequence that has the highest generation probability calculated as:

$$\bar{\sigma}(m) = \arg\max_{\sigma(m)} \sum_{Z} P(x_{\sigma[0]}, x_{\sigma[1]}, \ldots, x_{\sigma[m]}, Z|\lambda) \tag{17}$$

To choose the best sequence for MULM, we first find the best topic **c** that generates each lectures in the test set according to equation 4. After that, we select the sequence that has the highest likelihood

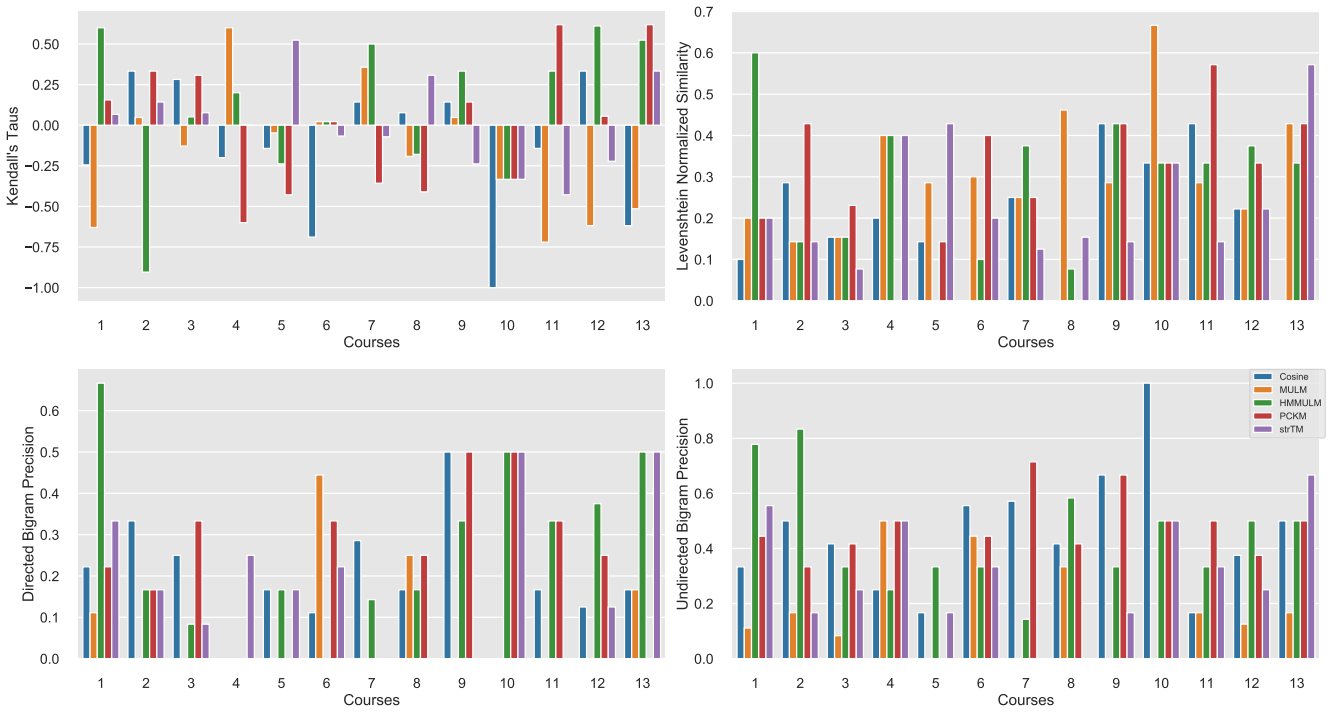


Figure 3: *The performance of different methods in **Task 3**: Predicting the whole sequence. All methods have comparable performance.*

based on the equation:

$$\begin{aligned}\bar{\sigma}(m) &= \arg\max_{\sigma(m)} P(C)P(X|C) \\ &= P(c_1)P(x_1|c_1)\sum_{i=2}^{|\sigma(m)|} P(c_i|c_{i-1})P(x_i|c_i)\end{aligned} \tag{18}$$

Since PCK-Means is a clustering method that minimizes the distance between lectures and clusters' centroids, we assign lectures x of the test set to the closest clusters z_i using Euclidean distances as shown in equation 19. Then, we select the sequence that maximizes the topic transitions between lectures in the sequence as well as minimizes the distance between adjacent lectures (see equation 20). The intuition behind that is to ensure the topic coherence between adjacent lectures and also reduces the gaps by minimizing the distance between them.

$$c(x) = \arg\min_{z_i} \|x - \mu_{z_i}\|^2 \tag{19}$$

$$\bar{\sigma}(m) = \arg\max_{\sigma(m)} \pi(c(x_1)) \sum_{i=2}^{|\sigma(m)|} A(x_{i-1}, x_i) - \|x_i - x_{i-1}\|^2 \tag{20}$$

As a baseline we accumulate the cosine similarity between adjacent lectures in the sequence and select the sequence in the permutations that has the highest similarity score to be the optimal sequence.

Table 2 summarizes the results of Task 1 for each method. We can notice that PCK-Means has the highest score in kendall's $\tau(\sigma)$ and Levenshtein normalized similarity in all datasets which indicates that PCK-Means has chosen the sequences that are very correlated to the ground truth sequences and need the minimal edits to be transformed to the ground sequences. However, PCK-Means only outperforms other models in the directed and undirected bigram precision in the Python dataset, indicating that it sometimes not able to capture the sequence between adjacent lectures.

Table 3: *The performance of **Task 2**: Predicting the next lecture. It is clear that HMMULM achienes the highest performance.*

	Accuracy		
Method	Python	SQL	ML
Cosine-Similarity	0.46	**0.56**	0.42
PCK-Means	0.45	0.49	0.47
MULM	0.41	0.34	0.37
HMMULM(Viterbi)	**0.52**	**0.56**	**0.60**
StrTM(Viterbi)	0.39	0.27	0.43

In general, it is clear that PCK-Means achieves the highest performance in most measures and almost in all the datasets. We think that combining the topic transitions with the Euclidean distance helps PCK-Means in finding the best sequence from the list of possible sequences.

6.2.3 *Task 2: Predicting The Next Lecture*

In task 2, each model predicts the next lecture given a sequence of lectures. We varied the length of the given sequence starting from one. As strTM and HMMULM are based on HMM, we utilized the Viterbi algorithm [20] to find the most probable sequence of hidden states or topics that generated the lectures in the given sequence. Then we greedily choose the next probable lecture in the sequence

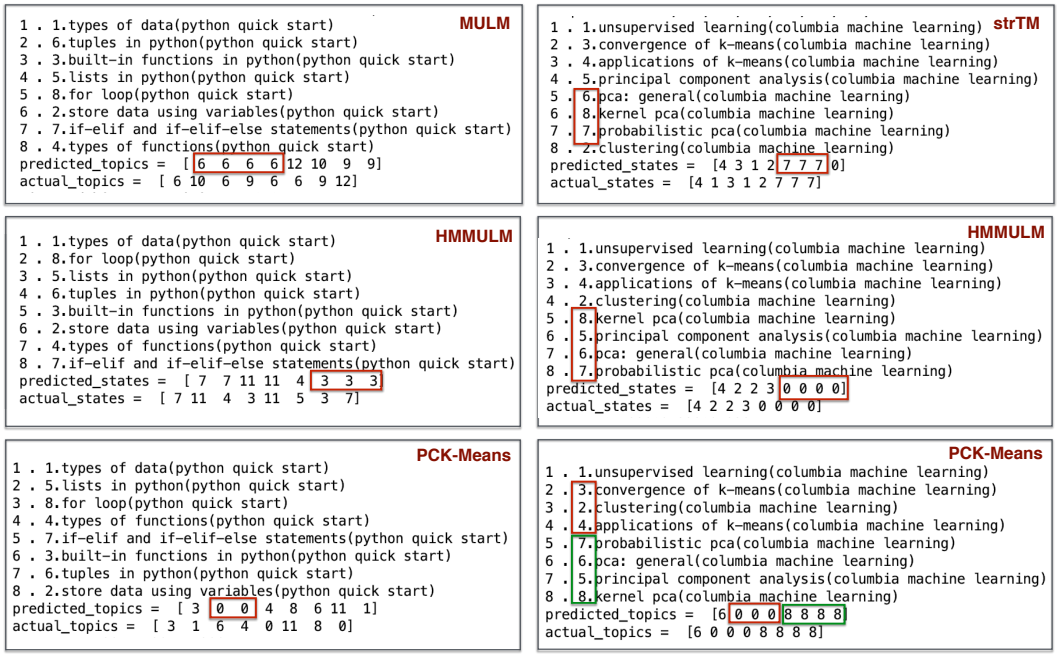


Figure 4: *Qualitative Analysis of Sequencing Task 3. (a) Examples of preferring self transition behaviour when selecting next lecture in the sequence, (b) Examples of the problem of sequencing adjacent lectures that cover the same topics.*

according to the equation:

$$\bar{x} = \arg\max_{x} P(z_i|z_{i-1})P(x|z_i) \tag{21}$$

Similar to task 1, for MULM and PCK-Means models we assign lectures to the best clusters using the equations 4 and 19 respectively. After that MULM greedily chooses the next lecture that maximizes the equation 21. On the other hand, PCK-Means model selects the next lecture that maximizes the topic transition and minimizes the distance with the last lecture in the given sequence. For the baseline, we use the cosine similarity where we choose the next lecture that has the highest similarity score with the last lecture in the given sequence.

Table 3 summarizes the results of Task 2 for each method. We can notice that HMMULM achieves the highest accuracy in all datasets. Using the Viterbi algorithm along with the learned topic transitions helps in capturing the most probable hidden states or topics that generate the given sequence of lectures. In addition, the topic transitions learned by HMMULM help in greedily pick the next lecture in the sequence. While StrTM also uses Viterbi algorithm similar to HMMULM, its accuracy scores were far less than HMMULM. We think the main reason for that due to the performance of the learned topic transitions as we explain in section 6.3.

6.2.4 Task 3: Predicting The Sequence

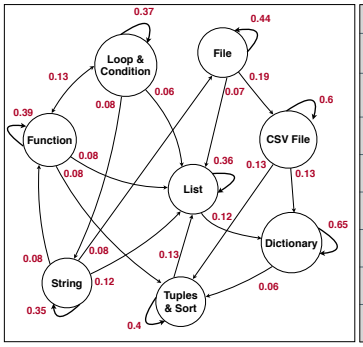
Task 3 is very similar to task 2 except that each method needs to find the whole sequence of given lectures where the first lecture in the sequence is given. Figure 3 depicts the results of Task 3.

As this task is considered the most challenging task, it is clear that there is no wining method. However, from the upper left graph that captures the Kendall's taus in Figure 3, we can notice that HMMULM has achieved a taus score ≥ 0.50 in four courses, PCK-Means has achieved the same score in only two courses, MULM and strTM in only one course, and Cosine method in non courses. For the Levenshtein normalized similarity, it is clear that all methods have comparable results. For the directed and undirected bigram precision, all methods have also comparable results except MULM. The reason is that MULM sometimes cannot complete the whole sequence because it only uses the greedy method which cannot complete the sequence in the case of the absence of the topic transitions required to sequence courses in the test set. In the case of other methods, they always find the whole sequence either because of the Viterbi algorithm used by HMMULM and strTM or due to the similarity or distance measures utilized by PCK-Means and Cosine methods.

In addition to quantitatively comparing the methods, we try to qualitatively evaluate the results by examining the generated sequences of each methods. In general, we found two common behaviour shared by all methods.

First, in most cases almost all the methods prefer self transition when they pick the next lecture in the sequence. For example, as shown in Figure 4 (a) , MULM, HMMULM, and PCK-Means select the next lecture that has the same topic as the current lecture.

Second, all methods cannot sequence lectures that belong to the same topic. In MOOCs, due to the short length of lectures, instructors sometimes explain the same topic using multiple lectures. As a result, it is hard to find the correct sequence of lectures that cover the same topic. For example, as shown in Figure 4 (b), the last four lectures of the course explain the "Principal Component Analysis algorithm" and hence strTM, HMMULM, and PCK-Means cannot predict the correct sequence of these lectures. In this case, we need to use other



Topics	Terms
String	string, index, quotes, strings, characters, quote, double, substring, character, eric, method.
Function	function, parameters, functions, arguments, return, parameter, formal, global, errors, def, scope.
File	file, files, open, directory, py, install, text, read, folder, line, lines.
CSV File	csv, phones, file, row, column, table, columns, delete, format, tabular, menu.
Loop & Condition	loop, true, false, condition, block, while, else, equal, statement, greater, boolean, infinite, elif.
Dictionary	dictionary, key, keys, dictionaries, pairs, counts, keyvalue, word, values, mapping, dict.
List	list, range, item, element, lists, index, elements, loop, sequence, items, strings.
Tuples & Sort	tuple, sorted, random, sort, tuples, sorting, list, reverse, lambda, function, order.

Classes	class, attributes, object, methods, teaching, classes, method, language, programming, objects, instance.
Prog. Language	programming, language, concepts, machine, hours, learn, tasks, tool, videos, calculations, languages.
List	list, accumulator, state, num, item, append, accumulate, items, object, position, mutate.
Not Clear	expression, area, largest, guess, recursive, root, expressions, divided, floating, pi, cube.
Not Clear	split, api, preview, join, space, image, white, comma, format, file, words.

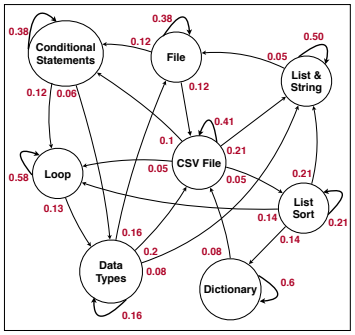
Figure 5: *Python topics and their transitions using **PCK-Means** method. The table represents the top terms in each topic.*

techniques to predict the sequence. One naive solution is to assume that all adjacent lectures the belong to the same topic as one atomic unit and we only need to sequence lectures that have different topics. Further investigation of solving the sequencing problem of lectures that belong to the same topic is left for future work.

6.3 Topic Transitions Examples

In this section, we present examples of topics and topic transitions learned by different methods. Due to space constraint, we present examples using Python dataset. We try to analyze the words with the highest probabilities in the word distributions of topics learned by each methods and manually mapped them to topic words or phrases. For instance, if the word distribution has the words: *list, range, items, index, and append*, then it is clear that this word distribution captures the topic "List". The word distributions with topic phrases of each topic learned by PCK-Means, MULM, HMMULM, and strTM methods in Python dataset are depicted in Figure 5, 6, 7, and 8 respectively. Since we have 13 topics in the Python dataset, we only visualize the topic transitions of a subset of these topics and depicted the transitions that have scores ≥ 0.05.

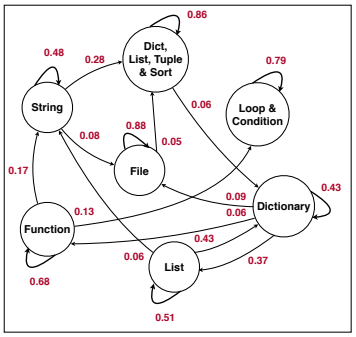
It is clear from the Figures that all models extract some useful topics where the top terms of each topic clearly explain the topic. However, PCK-Means has the best word distributions that clearly explain each topic followed by HMMULM and then MULM while strTM has the lowest performance. We also notice from the Figures that PCKMeans have extracted 11 useful topics with two topics that have unclear word distributions and cannot be mapped to any useful topics. In contrast, MULM has modeled 10 meaningful topics with three topics form noise and cannot be mapped to any topics. On the other hand, HMMULM and strTM capture 9 topics with four unclear topics that cannot be mapped to any phrase. In general, this finding indicates that PCK-Means has the best performance in modeling the topics of the courses in the Python dataset as it models more useful topics with clear word distributions. The results also indicate that strTM achieves the lowest performance because even though it captures the same number of meaningful topics as HMMULM, strTM has the lowest performance in the clarity of the word distributions.



Topics	Terms
CSV File	csv, quotes, file, string, format, row, quote, column, double, table, comma, columns.
File	file, function, open, lines, line, area, seconds, files, read, object, close.
Loop	loop, largest, while, infinite, loops, smallest, break, statement, blah, true, iteration.
Data Types	float, expression, integer, floating, type, int, types, floats, expressions, convert, decimal.
Conditional Statements	true, false, block, else, condition, equal, expression, greater, statement, username, elif, boolean.
Dictionary	dictionary, key, keys, dictionaries, pairs, counts, keyvalue, list, word, tuple, values.
List Sort	sorted, random, sorting, sort, list, programming, accumulation, language, loop, lost, lesson.
List & String	list, index, object, string, method, methods, class, item, position, strings, items.

Installing Python	file, install, directory, py, command, tool, windows, files, window, click, folder, ipython.
Prog. Language	programming, language, learn, concepts, tasks, scripts, practice, learning, videos, script, feel.
Not Clear	accumulator, list, guess, state, root, function, square, cube, dictionary, count, total.
Not Clear	turtle, floor, api, random, comments, calculations, module, program, errors, stack, machine.
Not Clear	list, tuple, function, element, range, loop, global, recursive, tuples, sequence, parameter.

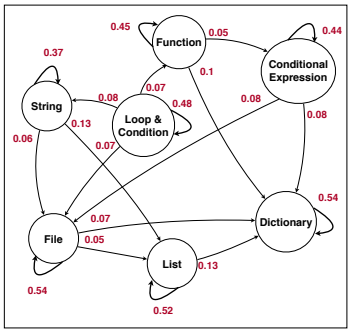
Figure 6: *Python topics and their transitions using **MULM** method. The table represents the top terms in each topic.*



Topics	Terms
String	string, index, character, strings, quotes, list, substring, characters, preview, location, slice.
Function	function, float, type, expression, return, parameter, parameters, integer, int, parentheses, string.
File	file, csv, files, phones, row, format, open, column, read, table, directory.
Loop & Condition	loop, true, false, while, condition, loops, equal, largest, block, infinite, else.
Dictionary	dictionary, list, accumulator, key, item, keys, count, counts, loop, state, num.
List	list, object, append, lists, items, position, lst, reference, strings, mutate, bound.
Dict, List, Tuple & Sort	dictionary, list, tuple, key, sorted, function, keys, sort, expression, item, sorting.

Classes	class, attributes, methods, instance, dictionary, method, turtle, classes, module, objects, concepts.
Prog. Language	language, programming, learn, script, data, learning, syntax, tasks, concepts, languages, scripts.
Not Clear	teaching, eric, code, cell, grade, install, button, messages, errors, error, month.
Not Clear	comments, calculations, floor, input, algorithms, computers, warm, web, city, hot, fast.
Not Clear	seconds, block, traceback, hours, blah, turtle, line, code, runs, times, execute.
Not Clear	kelvin, temperature, hours, fahrenheit, equal, program, step, error, loop, celsius, indent.

Figure 7: *Python topics and their transitions using **HM-MULM** method. The table represents the top terms in each topic.*



Topics	Terms
String	list, string, if, position, index, python, strings, quotes, character, number, single, type.
Function	function, return, code, parameter, expression, square, inside, functions, print, parameters, statement, arguments.
File	file, csv, files, open, line, read, list, row, lines, data, quotes, table.
Loop & Condition	if, loop, code, while, run, else, statement, block, equal, condition, true.
Dictionary	dictionary, key, list, function, keys, dictionaries, sorted, lists, sort, tuple, values.
List	list, lists, element, item, index, loop, function, if, count, strings, file.
Conditional Expression	true, if, false, else, equal, expression, code, boolean, block, greater, expressions, operators.

List & Dict	list, character, dictionary, item, loop, accumulator, variable, number, count, index, items.
Classes	list, class, tuple, function, method, set, data, tuples, index, methods, object.
Not Clear	loop, if, true, python, statement, largest, code, false, run, smallest, variable.
Not Clear	python, variable, string, type, if, integer, point, variables, code, floating, kind.
Not Clear	if, loop, guess, root, machine, start, simply, number, answer, times, cube.
Not Clear	data, dictionary, language, kelvin, keys, programming, table, row, column, dictionaries, columns.

Figure 8: *Python topics and their transitions using **strTM** method. The table represents the top terms in each topic.*

As shown in the Figures 5, 6, 7, and 8, the meaningful topics extracted by all methods are very similar with some variations. For example, while PCK-Means and MULM separate "*Files*" and "*CSV Files*", HMMULM and strTM combines them into one topic. In addition, while PCK-Means and HMMULM combines "*Loops & Condition*", MULM differentiates between them. StrTM, on the other hand, has both "*Loops & Condition*" and "*Conditional Statements*."

Regarding the topic transitions, it is clear that all models capture self transitions with the topic and itself. This indicates that, in MOOCs, instructors used multiple lectures to explain the same topic. However, HMMULM gave higher probability to self transitions compared to other methods. We can notice from the Figures that there are some consensus between all methods on some transitions between topics such as: "*List*"→ "*Dictionary*" and "*String*"→ "*File*." There are also some variations of topic transitions between different models. For instance, wile PCK-Means, HMMULM, and strTM have a transition between "*String*"→ "*List*", MULM combines these two topics into one topic or cluster. Another variation is that, PCK-Means, strTM and MULM have a transition "*Loop & Condition*"→ "*String*", whereas HMMULM misses this transition.

In general, all methods captures useful topics with clear word distributions. Regarding the topic transitions, all methods capture self transitions and also have some consensus on some transitions. There are also some variations between methods and these differences due to how each method identify topics of each lecture. Improving the modeling of topics and the mapping between lectures and topics clearly would improve the quality of the topic transition maps.

7. CONCLUSION

In this paper, we introduce the Topic Transition Map which is a general structure that models the content of MOOCs as topics, where each lecture is mapped to a topic, and captures the transition between topics. It models the various ways of how instructors organize topics in order to construct the study plan of their courses. We investigate four different methods to construct the Topic Transition Map: PCK-Means, MULM, HMMULM, and strTM. PCK-Means and MULM separately cluster lectures into topics and then learn the transitions between topics, by leverage the sequences of lectures in different courses. In contrast, HMMULM and strTM assume first order Markov property among latent topics and hence jointly learn topics and their transitions. While the three model, MULM, HMMULM, and strTM are probabilistic models, PCK-Means is distance-based clustering algorithm that incorporates some constraints to guide the clustering process.

We evaluated the generated topic transitions from various methods using three different tasks: 1) determining the correct sequence, 2) predicting the next lecture, and 3) predicting the sequence of lectures. Our evaluation revealed that PCK-Means achieves the highest performance in determining the correct sequence while HMMULM outperforms other methods in the task of predicting the next lecture. Since the task of predicting the whole sequence of lectures is considered the most challenging task, there was no winning method and all methods have comparable performance with MULM has the lowest performance as it sometimes fails to predict the whole sequence. We also visualize the the Topic Transition Maps generated by different methods to qualitatively evaluate the resulted maps. We found that PCK-Means has extracted more meaningful topics with the best word distributions that clearly explain each topic.

In the future, we plan to explore incorporating Topic Transition Map with concept dependency relations and examine if this can solve the problem of sequencing lectures that belong to the same topic. Further, we aim to combine different methods such as PCK-Means and HMMULM in order to improve the accuracy of the Topic Transition Map and hence improving the performance of the sequencing tasks. Finally, we plan to apply our work on other domains such as traditional University courses or educational books. To do that, we need to investigate how to divide long lectures or book sections into segments where each segment is mapped to one topic.

8. REFERENCES

[1] R. Agrawal, B. Golshan, and E. Papalexakis. Toward data-driven design of educational courses: A feasibility study. *JEDM-Journal of Educational Data Mining*, 8(1):1–21, 2016.

[2] F. ALSaad and A. Alawini. Unsupervised approach for modeling content structures of moocs. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, pages 18–28, 2020.

[3] F. ALSaad, A. Boughoula, C. Geigle, H. Sundaram, and C. Zhai. Mining mooc lecture transcripts to construct concept dependency graphs. In *Proceedings of the 11th International Conference on Educational Data Mining*, pages 467–473. EDM, 2018.

[4] S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 333–344. SIAM, 2004.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[6] D. Chaplot and K. R. Koedinger. Data-driven automated induction of prerequisite structure graphs. In *Proceedings of the 9th International Conference on Educational Data Mining*, pages 318–323. EDM, 2016.

[7] W. Chen, A. S. Lan, D. Cao, C. Brinton, and M. Chiang. Behavioral analysis at scale: Learning course prerequisite structures from learner clickstreams. In *Proceedings of the 11th International Conference on Educational Data Mining*, pages 66–75. EDM, 2018.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39 (1):1–22, 1977.

[9] L. Du, J. K. Pate, and M. Johnson. Topic models with topic ordering regularities for topic segmentation. In *2014 IEEE International Conference on Data Mining*, pages 803–808. IEEE, 2014.

[10] L. D. Fink. *Creating significant learning experiences: An integrated approach to designing college courses*. John Wiley & Sons, 2013.

[11] R. M. Gagne and L. J. Briggs. *Principles of instructional design.* Holt, Rinehart & Winston, 1974.

[12] A. Gruber, Y. Weiss, and M. Rosen-Zvi. Hidden topic markov models. In *Artificial intelligence and statistics*, pages 163–170. PMLR, 2007.

[13] T. Hofmann. Probabilistic latent semantic analysis. *arXiv preprint arXiv:1301.6705*, 2013.

[14] C. Liang, J. Ye, Z. Wu, B. Pursel, and C. L. Giles. Recovering concept prerequisite relations from university course dependencies. 2017.

[15] H. Liu, W. Ma, Y. Yang, and J. Carbonell. Learning concept

graphs from online educational data. *Journal of Artificial Intelligence Research*, 55:1059–1090, 2016.

[16] R. Manrique, J. Sosa, O. Marino, B. P. Nunes, and N. Cardozo. Investigating learning resources precedence relations via concept prerequisite learning. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 198–205. IEEE, 2018.

[17] L. Pan, C. Li, J. Li, and J. Tang. Prerequisite relation learning for concepts in moocs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1447–1456, 2017.

[18] A. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment*, 3(1-2):566–577, 2010.

[19] G. Pfundstein. *Hidden markov models with generalised emission distribution for the analysis of high-dimensional, non-euclidean data*. PhD thesis, Institut für Statistik, 2011.

[20] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[21] R. Scheines, E. Silver, and I. M. Goldin. Discovering prerequisite relationships among knowledge components. In *EDM*, pages 355–356, 2014.

[22] S.-s. Shen, H.-y. Lee, S.-w. Li, V. Zue, and L.-s. Lee. Structuring lectures in massive open online courses (moocs) for efficient learning by linking similar sections and predicting prerequisites. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[23] A. Vuong, T. Nixon, and B. Towle. A method for finding prerequisites within a curriculum. In *EDM*, pages 211–216, 2011.

[24] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.

[25] H. Wang, D. Zhang, and C. Zhai. Structural topic model for latent topical structure analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1526–1535, 2011.

[26] S. Wang, A. Ororbia, Z. Wu, K. Williams, C. Liang, B. Pursel, and C. L. Giles. Using prerequisites to extract concept maps fromtextbooks. In *Proceedings of the 25th acm international on conference on information and knowledge management*, pages 317–326, 2016.

[27] K. WAUTERS, P. DESMET, and W. VAN DEN NOORTGATE. Acquiring item difficulty estimates: a collaborative effort of data and judgment. In *EDM 2011 4 th International Conference on Educational Data Mining*, page 121.

[28] J. Xu, J. Liu, and K. Araki. A hybrid topic model for multi-document summarization. *IEICE TRANSACTIONS on Information and Systems*, 98(5):1089–1094, 2015.

[29] Y. Yang, H. Liu, J. Carbonell, and W. Ma. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 159–168. ACM, 2015.

[30] B.-J. Yoon. Hidden markov models and their applications in biological sequence analysis. *Current genomics*, 10(6): 402–415, 2009.

[31] C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 743–748, 2004.

[32] S. Zheng, M. B. Rosson, P. C. Shih, and J. M. Carroll. Understanding student motivation, behaviors and perceptions in moocs. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, pages 1882–1895. ACM, 2015.

Can Feature Predictive Power Generalize? Benchmarking Early Predictors of Student Success across Flipped and Online Courses

Mirko Marras
EPFL
mirko.marras@acm.org

Julien Tuân Tu Vignoud
EPFL
julien.vignoud@epfl.ch

Tanja Käser
EPFL
tanja.kaeser@epfl.ch

ABSTRACT

Early predictors of student success are becoming a key tool in flipped and online courses to ensure that no student is left behind along course activities. However, with an increased interest in this area, it has become hard to keep track of what the state of the art in early success prediction is. Moreover, prior work on early success prediction based on clickstreams has mostly focused on implementing features and models for a specific online course (e.g., a MOOC). It remains therefore under-explored how different features and models enable early predictions, based on the domain, structure, and educational setting of a given course. In this paper, we report the results of a systematic analysis of early success predictors for both flipped and online courses. In the first part, we focus on a specific flipped course. Specifically, we investigate eight feature sets, presented at top-level educational venues over the last few years, and a novel feature set proposed in this paper and tailored to this setting. We benchmark the performance of these feature sets using a RF classifier, and we provide and discuss an ensemble feature set optimized for the target flipped course. In the second part, we extend our analysis to courses with different educational settings (i.e., MOOCs), domains, and structure. Our results show that (i) the ensemble of optimal features varies depending on the course setting and structure, and (ii) the predictive performance of the optimal ensemble feature set highly depends on the course activities.

Keywords

Flipped Classroom, MOOC, Success Prediction, Early Warning, Clickstream, At-Risk Students, Learning Analytics.

1. INTRODUCTION

An increasing number of universities are now running blended courses that combine traditional lectures with online instruction, providing educational models tailored to the needs of our society [20]. A popular instructional strategy to enable blended learning is represented by *flipped classrooms*, where students complete *pre-class activities* before attending face-to-face lessons [18]. Recent studies have shown the positive impact and dependency of this strategy on student-centered variables such as self-efficacy and self-regulation [22, 17, 6, 19]. Pre-class activities usually consist in watching videos and completing quizzes part of *Massive Open Online Courses* (MOOCs) used as supplementary material [27]. Each week, students are asked to perform these pre-class activities and to then complete exercises and have discussion in class. Pre-class activities are fundamental for the success of flipped courses [12, 21, 28]. However, students often lack skills, time, and motivation to regulate their pre-class activity; as a consequence, they may experience difficulties in class and end up failing the course [10, 14]. To ensure that no learner is left behind, *Early Success Predictors* (ESPs) are becoming crucial to support instructors in identifying and timely acting upon risk factors of failing a course.

So far, there are few studies on analyzing student success in flipped courses based on pre-class activities. For instance, Jovanovic et al. [9, 8] clustered interaction sequences in pre-class clickstreams to identify learning strategies, showing how strategy-based student profiles differ in course grades. Beatty et al. [2] found that frequency counts of video usage are often correlated with course grades in flipped classrooms. In blended, but not flipped settings, Akpinar et al. [1] showed that student's strategy counts, with strategies modelled as clickstream event n-grams, are indicative of course homework grades. Wan et al. [25, 26] trained gradient boosting classifiers on an extensive set of clickstream-based features to identify at-risk students in a small private online course delivered in hybrid mode. They also analyzed the importance of the features, finding that the time spent in online activities and the stability of time distribution during weeks have the highest importance in that course. To the best of our knowledge, no prior work on flipped courses specifically focused on ESPs.

Conversely, there is a large body of research on success prediction for fully online courses (e.g., MOOCs). A multitude of feature sets have been extracted from clickstreams for this purpose. Recent work proposed video-counting (e.g., number of videos viewed per week, rewinds, fastforwards, pauses, and plays, and the fractional and total amount of time played and paused for videos) and session-based (e.g., number of sessions, mean and standard deviation of the time for all sessions and between sessions) features [4, 13]. These features were fed into different commonly used classifiers

Mirko Marras, Julien Tuan Tu Vignoud and Tanja Käser "Can Feature Predictive Power Generalize? Benchmarking Early Predictors of Student Success across Flipped and Online Courses". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 150-160. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

(e.g., Logistic Regression, Naive Bayes, Decision Tree, RF, and Neural Networks) to predict success in weekly assignments or in the entire specific MOOC. In [3], several features that measure intra-course, intra-week and intra-day regularity in video watching were proposed, and their correlation with the course grade was shown. Other researchers leveraged attendance rates, usage rates, and watching ratios [7, 15]. Specifically, they explored how the difference in these indicators affects academic performance, showing that students whose indicators are high are more likely to graduate on schedule. More fine-grained features on video usage (e.g., total video views, mean and standard deviation of the proportion of videos watched, re-watched, and interrupted per week, and the frequency and total number of all video actions and of each type of video action) were proposed in [11]. The authors clustered students according to their watching behavior and found that such a behavior is representative of course performance. Similarly, Mubarak et al. [16] extracted implicit features from video-clickstream data, and investigated the extent to which neural networks fed with those features can predict weekly students' performance. For an extensive discussion on success prediction in MOOCs, we recommend this survey [5].

The above features and classifiers, however, are designed for fully-online learning contexts, such as MOOCs. Despite clear connections, there are essential aspects which distinguish flipped courses from MOOCs. First of all, flipped course data includes relatively few students. A large part of the learning activity happens offline and cannot be tracked, leading to data only on course segments. Flipped courses generally have also an intense instructor guidance and performance on them has direct impact on the academic portfolio. As a motivating example, we consider a flipped course on Linear Algebra later described in this paper and the regularity features proposed for MOOCs in [3]. They quantify students' time regularity by considering their activities over the course (e.g., studying at the same days of the week). Boroujeni's study revealed that the final grade in the MOOC is correlated with two intra-week regularity measures and the periodicity of day hour and week hour ($.46 < c < .7$, $p < 0.001$). Conversely, the same features resulted to have no correlation with the final grade in the above flipped course ($.0 < c < .1$, $p < 0.001$). Therefore, it remains unexplored whether existing features and classifiers for MOOCs generalize to different educational settings (e.g., flipped classrooms), and to what extent the feature importance varies according to the topic, structure, and educational setting of the course.

The contribution of this paper is two-fold: we tackle the problem of ESPs in flipped classroom settings[1], and we provide an extensive analysis and benchmark of classifiers and features for early success prediction across different types of courses, namely MOOCs and flipped courses. A schematic overview of our analysis in this paper is shown in Figure 1.

In a first step, we propose a novel feature set for early success prediction in flipped courses. Our feature set measures students' alignment, anticipation, and strength in quiz and video usage. We benchmark our new feature set using a Random Forest (RF) classifier against eight feature sets presented in previous work on success prediction in online courses. We retrieved these feature sets by systematically scanning the recent papers published at major educational venues (e.g., EDM, AIED, etc.) and reproducing the features based on the relevant papers. Our results on data of 214 students enrolled in a linear algebra flipped course show that the novel feature set outperforms all previously suggested feature sets. We also show that predictive performance can be increased by selecting the optimal features from the ensemble of all feature sets.

In a second step, we extend our analysis to further courses along three dimensions: domain, structure, and educational setting. We compute the early predictive performance again using a RF classifier for three additional courses: a flipped course on functional programming (where pre-class activities include videos only), a MOOC on linear algebra (including video and quiz activities), and a MOOC on functional programming (including video activities only). For each course, we select the optimal features from the ensemble of feature sets (eight feature sets from prior work and one novel feature set from this paper) as input features for the RF classifier. Our results show that the structure of the course significantly influences performance. Predictive performance for courses including quizzes is much higher than for courses including only videos. Furthermore, we also show that while there is some overlap between the optimal features across courses, the importance of the features highly depends on the setting and structure of the course.

2. EARLY PREDICTION FORMULATION

The problem addressed in this paper can be framed into a time series classification task that relies on clickstreams to predict student success in a course. For clarity and reproducibility, we present and formalize the addressed problem.

Course. Early success predictions are provided in the context of a course (e.g., a MOOC or a course run in a flipped classroom setting). In what follows, we hence mathematically define fundamental concepts, such as the course schedule, the learning objects, and their properties. Specifically, we consider a set of students $\mathbb{U}$ who are enrolled in a course c part of the online educational offering $\mathbb{C}$. Each course $c \in \mathbb{C}$ has a pre-defined schedule $\mathbb{S}_c$ consisting of $N = |\mathbb{S}_c|$ online activities, such that $\mathbb{S}_c = \{s_1, \ldots, s_N\}$. We assume that each online activity s_j included in the course schedule is represented by a tuple (o_j, t_j), consisting of learning object $o_j \in \mathbb{O}$ and its corresponding completion deadline for students $t_j \in \mathbb{R}^+$, modelled as a timestamp. Each learning object $o \in \mathbb{O}$ is characterized by descriptive properties denoted with an M-dimensional vector $f_o = (f_1, \ldots, f_M)$ over a set of features $\mathbb{F} = \{\mathbb{F}_1, \ldots, \mathbb{F}_M\}$ that vary according to the type of the learning object (e.g., the duration for a video or the maximum grade for a quiz). Specifically, each feature $\mathbb{F}_j \in \mathbb{F}$ can be envisioned as a set of discrete or continuous values describing an attribute of a learning object o, $f_{o,j} \in \mathbb{F}_j$ for $j = 1, \ldots, M$. Our study in this paper assumes that learning objects can be either videos or quizzes, but the notation can be easily extended to other types (e.g., forum posts or readings). The type of a learning object $o \in \mathbb{O}$ is returned by a function $type : \mathbb{O} \rightarrow \{video, quiz\}$.

[1] https://github.com/d-vet-ml4ed/flipped-classroom

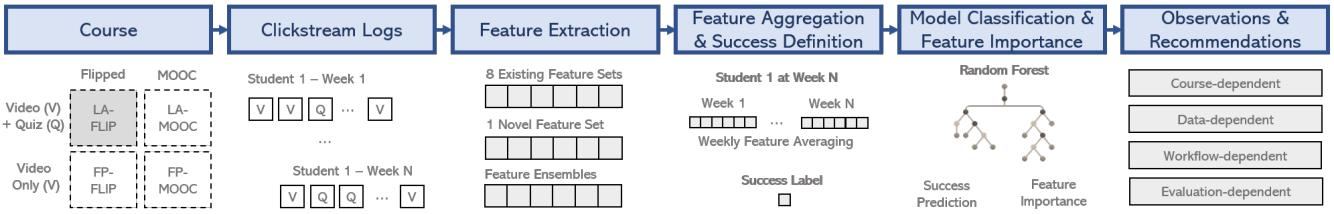


Figure 1: Our Framework. We first analyze a flipped course with videos and quizzes, then investigate differences between courses in flipped and MOOC settings and with videos only and videos plus quizzes. Eight state-of-the-art feature sets, a novel feature set, and feature ensembles are computed for each student and each week of the course. Weekly features are averaged and a success label is attached, according to the course type. Classification is performed using a Random Forest. Observations and recommendations on the predictive power of features are provided for each course setting, highlighting open challenges.

Based on common log data collected by educational platforms, we assume that learning objects of type *video*, denoted as $\mathbb{O}^{video} = \{o \in \mathbb{O} \,|\, \text{type}(o) = video\}$, are described by properties associated to the video duration in seconds as $\mathbb{F}^{video} = (\text{duration} \in \mathbb{R}^+)$. Learning objects of type *quiz*, denoted as $\mathbb{O}^{quiz} = \{o \in \mathbb{O} \,|\, \text{type}(o) = quiz\}$, are characterized by descriptive properties that model the maximum grade students can achieve in that quiz as $\mathbb{F}^{quiz} = (\text{maxgrade} \in \mathbb{R}^+)$. For convenience, we use superscripts to denote a descriptive property of a learning object. For instance, the duration of a video $o \in \mathbb{O}^{video}$ can be referred to as $o^{duration}$. The same notation applies to other descriptive properties.

Interaction. Students enrolled in an online course interact with the learning objects included in the course schedule, generating a time-wise clickstream. We denote a clickstream in a course $c \in \mathbb{C}$ for a student $u \in \mathbb{U}$ as a time series I_u, such that $I_u = \{i_1, ..., i_K\}$, with $K \in \mathbb{N}$ (e.g., a sequence of video plays and pauses, quiz submissions, and so on). We leave these definitions very general on purpose, in particular allowing the length of each time series to differ, since our models are inherently capable of handling this. Likewise, we neither enforce nor expect all time series to be synchronized, i.e. being sampled at the same time, but rather we are fully agnostic to non-synchronized observations. This configuration is common in educational time series. We assume that each interaction i_j is represented as a tuple (t_j, a_j, o_j, d_j), consisting of a timestamp $t_j \in \mathbb{R}^+$, the action $a_j \in \mathbb{A}$ performed by the student (e.g., play or pause), the learning object $o_j \in \mathbb{O}$ involved in the action a_j (e.g., a certain video or quiz), and an L-dimensional descriptive vector $d_j = (d_1, \ldots, d_L)$ over a set of features $\mathbb{D} = \{\mathbb{D}_1, \ldots, \mathbb{D}_L\}$. These descriptive vectors are used to append relevant information to an action a_j performed at time t_j, such as the current video time when the action occurred or the grade received by the student on a quiz. Based on the type of the learning object $o \in \mathbb{O}$, the student can perform different actions $\mathbb{A}$. We assume that video interactions, denoted by $\{i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \,|\, \text{type}(o_j) = video\}$, are limited to actions $a_j \in \mathbb{A}^{video} = \{Load, Play, Pause, Stop, SpeedChange, Seek\}$. These actions are derived from those commonly allowed to students in online educational platforms. Conversely, quiz interactions, denoted by $\{i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \,|\, \text{type}(o_j) = quiz\}$, include actions $a \in \mathbb{A}^{quiz} = \{\text{Submit}\}$.

In online educational platforms, clickstream interactions include a payload with additional information beyond the timestamp, the action, and the involved learning object. For instance, if a student submits a quiz, the resulting interaction includes also the grade assigned by the system to the student's quiz. Our notation models each dimension $\mathbb{D}_l \in \mathbb{D}$ of a clickstream interaction as a set of discrete or continuous values describing the interaction $i_j \in \mathbb{I}_u$, $d_{j,l} \in \mathbb{D}_l$ for $l = 1, \ldots, L$. Specifically, we assume that interactions involving base video actions $\{i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \,|\, a_j \in \{Load, Play, Pause, Stop\}\}$ include descriptive properties associated to the current video time the interaction occurred, i.e. $\mathbb{D}^{Base} = (\text{current-time} \in \mathbb{R}^+)$. Interactions involving a speed change in a video, denoted as $\{i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \,|\, a_j \in \{SpeedChange\}\}$, are characterized by descriptive properties associated to both the old and the new speed the video has been and will be watched, i.e. $\mathbb{D}^{SpeedChange} = (\text{oldspeed} \in \mathbb{R}^+, \text{newspeed} \in \mathbb{R}^+)$. Interactions generated by students while seeking the video backward or forward, denoted as $\{i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \,|\, a_j \in \{Seek\}\}$, are modelled by descriptive properties related to the previous and current video time the student moved on, i.e. $\mathbb{D}^{Seek} = (\text{oldtime} \in \mathbb{R}^+, \text{newtime} \in \mathbb{R}^+)$. Finally, submit interactions generated in quiz activities, denoted as $\{i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \,|\, a_j \in \{Submit\}\}$, include descriptive properties on the grade assigned to the quiz answer and the progressive number of the attempt made on that quiz, i.e. $\mathbb{D}^{Submit} = (\text{grade} \in \mathbb{R}^+, \text{subnum} \in \mathbb{R}^+)$, with $\text{grade} \in [0, 1]$.

For convenience, we denote as $\mathbb{I}_u^t$ the clickstream including interactions $i_j \in \mathbb{I}_u$, such that $t_j < t \; \forall t_j \in \mathbb{I}_u^t$, namely those occurred before time t. Similarly, since online activities in MOOCs and flipped courses are organized on a weekly basis, t_w identifies the time t where the course week w ends. For instance, the clickstream of user u generated till the end of the second week can be denoted as $\mathbb{I}_u^{t_2}$.

Success Label. Once interactions are modelled, we need to associate a success label according to the final grade the corresponding student has received for that course. We consider a dataset $\mathbb{G}$ to consist of tuples, i.e. $\mathbb{G} = \{(I_{u_j}, y_{u_j})\}$, where I_{u_j} denotes the interactions of student u_j and $y_{u_j} \in \{0, 1\}$ the pass-fail label or the above-below average grade label.

Feature Extraction. Machine-learning models rarely receive raw interaction sequences, as so we abstract such interactions through a feature extraction step. Given the interactions $\mathbb{I}_u^{t_w} \subset \mathbb{I}_u \in \mathbb{I}$, generated by student u till the course week $w \in \mathbb{N}$, we produce fixed-length representations in

$\mathbb{H} \subset \mathbb{R}^{w \times H}$, where $H \in \mathbb{N}$ is the dimensionality of the feature set. Therefore, we assume that H-dimensional vectors are extracted for each week. For instance, if the feature set includes "number of sessions" and "number of clicks", feature vectors of size $H = 2$ are extracted each week. Formally, the extraction process is denoted as $\mathcal{H} : \mathbb{I} \rightarrow \mathbb{H}$, from interactions to features.

Model. Given the dataset $\mathbb{G}$ with interactions - success label pairs, an early success predictor $\mathcal{E}$ aims to predict the success label y_{u_j} associated to the interactions $\mathbb{I}_{u_j}$. Formally, this operation can be abstracted as a function $\widetilde{y}_{u_j} = \mathcal{E}(\mathbb{I}_{u_j}|\theta)$, where $\widetilde{y}_{u_j}$ denotes the predicted label, θ denotes the model parameters, and $\mathcal{E}$ denotes the predictive function that maps interactions $\mathbb{I}_{u_j}$ to the predicted label $\widetilde{y}_{u_j}$ according to θ.

Objective Function. Hence, training an early success predictor $\mathcal{E}$ with interactions - success label pairs till course week w becomes an optimization problem, aimed to find model parameters θ that maximize the expectation on the following objective function (i.e., predicting the correct success label, given the interactions) on a dataset $\mathbb{G}$:

$$\tilde{\theta} = \underset{\theta}{\operatorname{argmax}} \underset{(\mathbb{I}_u, y_u) \in \mathbb{G}}{\mathbb{E}} y_u = \mathcal{E}(\mathcal{H}(\mathbb{I}_u^{t_w}) \mid \theta) \tag{1}$$

In this paper, we focus on *feature extraction*, which formally results in the operationalization of the function $\mathcal{H}$.

3. REPRESENTATIVE FEATURE SETS

To make sure that our work is not only based on individual examples of published research, we systematically scanned the proceedings of conferences and journals for relevant papers in a manual process. In our analysis, we considered papers that appeared in the last years in the top educational technology conferences (e.g., LAK, EC-TEL, AIED, and EDM) and journals (e.g., IEEE TLT, Springer EIT, Journal of Learning Analytics). We considered a paper to be relevant if it (a) proposed a novel feature set for course success analysis, and (b) focused on the context of online courses or courses with online activities. Papers on other tasks, e.g., prediction of affective state or conceptual understanding, or other educational contexts, e.g., interactive simulations or games, were not considered. Moreover, papers with highly overlapping feature sets were filtered, and the paper with the most extensive set was used as representative. Finally, eight papers were included in our study.

In a next step, we reproduced the feature sets described in the above papers. Our approach was to rely as much as possible on the artifacts provided by the authors themselves, i.e., their source code and the descriptions included into the papers. In theory, it should be possible to reproduce published results using only the technical descriptions in the papers. In reality, there are many tiny implementation details with an impact on experiments. Overall, we could reproduce with reasonable assumptions all eight feature sets based on the relevant papers. In what follows, we give a description of each feature set included in our study.

AkpinarEtAl. This feature set consists of consecutive subsequences of n clicks extracted from the session clickstreams of a blended course [1]. In addition to sub-sequences, the authors considered four features related to the number of clicks, the number of session clickstreams, and attendance information. Note that in comparison to the original paper, we extract sequences from a different set a raw events, namely only videos and quizzes (e.g., no events on forums). Hence, in our case the feature set has a size of $|\mathbb{A}^{video} \cup \mathbb{A}^{quiz}|^n$ features per student. Since we expect short patterns to be un-interpretable and particularly long patterns to be rare, we choose $n = 3$ for our analyses.

BoroujeniEtAl. This feature set was originally used to measure to what extent MOOC students are regular in their study patterns [3]. Specifically, it is considered whether students study on certain hours of the day, day(s) of the week or similar weekdays. Other features monitor whether students have the same distribution of study time among weekdays over weeks, particular amount of study time on each weekday, and finally to what extent a student follows the schedule of the course. This set includes 9 features per student. Other papers proposed similar regularity features [23, 24, 8, 9, 2]. We limit our analysis to the feature set listed in [3], as in our first experiments it exhibited the best predictive power (among papers focusing on regularity features).

ChenCui. The feature set presented in this paper [4] includes click countings from a mandatory undergraduate course run through Moodle. Features include the number of total clicks and of clicks on campus, the ratio of on-campus to off-campus clicks, the number of online sessions (with average and standard deviation), standard deviation of time between online sessions, number of clicks during weekdays or weekends, ratio of weekend to weekday clicks, and the number of clicks for each type of module (e.g., assignment, forum, and quiz). To accomodate the scenario presented in Section 2, our study does not cover the features not easily generalizable to different types of online courses: the number of clicks on campus, the ratio of on-campus to off-campus clicks, the number of clicks for modules file, forum, report system. We therefore obtain a feature set of size 13 for each student.

LalleConati. This paper [11] focuses again on MOOCs. The presented feature set is composed by video interaction features at two levels of granularity. Features on video views include the total number of videos views (both watches and rewatches), in addition to the average and standard deviation of the proportion of videos watched, re-watched, and interrupted per week. On the other hand, features on actions performed within the videos include the frequency and total number of all performed video actions, frequency of video actions for each type of video action, and the average and standard deviation duration of video pauses, seek lengths, and so on. This feature set has a size of 22 per student.

LemayDoleck. The next paper [13] is also focused on MOOCs. Presented features include the number of videos watched per week, the average time fraction paused, played or spent watching, the average and standard deviation of the playback rate, and the total number of rewinds, pauses, and fast-forwards. Note that this feature set includes only video-related measures, resulting in vectors of size 10 per student.

MbouzaoEtAl. In this MOOC paper [15], the authors introduce three novel features, namely attendance rate, uti-

lization rate, and watching ratio. The attendance rate of a student on a given week is the number of videos the student played over to the total number of videos in that week. The utilization rate is the proportion of video play time activity of the student over the sum of video lengths for all videos on that week. Finally, the watching ratio is defined as the product between the two former features. This 3-sized feature set has been tested in MOOCs, extending an already-existing feature set [7].

MubarakEtAl. This paper [16] is primarily focused on implicit features about video-usage behavior in MOOCs. Composed by 13 features, this set covers fine-grained characteristics, such as the percentage of the video the learner watched not counting repeated segments, the amount of real time the learner spent watching the video (i.e. when playing or pausing) compared to the video duration, and the sum of times a learner viewed a video in its entirety.

WanEtAl. This set was designed for a small private online course [26]. Features measure the online learning time, the strength of the learner's engagement in forums and weekly assignments, the extent to which students attempt to do the homework soon, as examples. Table 1 and 2 in the cited paper provide further details. Given that we do not cover forum interactions, our study does not consider forum features. Finally, this set includes 14 features per student.

4. EARLY PREDICTORS IN FLIPPED COURSE SETTINGS

In this section, we first present a novel feature set for flipped courses, based on alignment, anticipation, and strength in content usage. We then describe the experimental setup and results aimed to assess to what extent the feature sets (including ours) are predictive of student success.

4.1 Our Feature Set

The feature sets presented so far mainly tackle video-related features and/or consider only low-level features, with only a few of them including features related to quizzes or assignments. Considering that predicting the success of a student based on clickstream data only is a challenging task per sé, we believe that limiting features to those extracted from videos may result in inferior predictive performance. We therefore suggest a number of additional features assessing students' knowledge and alignment with the course schedule.

Competency Strength is defined as the average of the inverse number of submissions for a quiz, weighted by the highest grade achieved by the student on that quiz. Given the inverse term, the value of this feature decreases when the student attempts the quiz multiple times and if the grade achieved by the student on the last attempt is not the highest-possible one. Hence, good-performing students may use few attempts and reach the maximum quiz grade fast (value close to 1). Students struggling with the material may attempt the quiz many times and not reach the maximum grade (value close to 0). Given a student u and the week w of the course, this feature is computed as:

$$\frac{1}{|Q_u|}\sum_{q\in Q_u}\frac{1}{Q_u^q}\,max(G_u^q) \tag{2}$$

where:

- $Q_u = \{o_j | i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \cap \text{type}(o_j) = quiz \cap t_j \leq t_w\}$ are the quizzes taken by student u till week w.
- $Q_u^q = |\{i_j | i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \cap o_j = q \cap t_j \leq t_w\}|$ is the number of attempts a student had on quiz q.
- $G_u^q = \{d_j^{\text{grade}} | i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \cap o_j = q \cap t_j \leq t_w\}$ is the set of grades a student got on quiz q.

Competency Alignment is defined as the number of quizzes the student received the maximum grade until week w, divided by the total number of quizzes scheduled for the period of consideration. Good-performing students may receive the maximum grade in all quizzes for the period of consideration (value close to 1); low-performing students may be behind the schedule and pass fewer quizzes than those proposed (value close to 0). Given a student u and the week w of the course, this feature is computed as:

$$\frac{|Q_u^{\text{pass}} \cap S^{leq(t_w)}|}{|S^{leq(t_w)}|} \tag{3}$$

where:

- $Q_u^{\text{pass}} = \{o_j | i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \cap \text{type}(o_j) = quiz \cap d_j^{\text{grade}} = o_j^{\text{maxgrade}}\}$ is the set of quizzes the student u received the maximum grade until week w.
- $S^{leq(t_w)} = \{o_j \in \mathbb{O} | (o_j, t_j) \in \mathbb{S}_c \cap type(o_j) = \text{quiz} \cap t_j \leq t_w\}$ is the set of quizzes to complete by week w.

Competency Anticipation is defined as the number of quizzes attempted by the student among those in subsequent weeks of the current week of study. This feature can be seen as a proxy of the learning propensity of a student. For instance, if a quiz is scheduled to be solved in subsequent weeks, we expect that good-performing students try them earlier, anticipating the deadline stated in the platform (value close to 1). Low-performing students may delay the consumption of quizzes across weeks or even towards the end of the course (value close to 0). Given a student u and the week w of the course, this feature is computed as:

$$\frac{|Q_u \cap S^{gt(t_w)}|}{|S^{gt(t_w)}|} \tag{4}$$

where Q_u is the set of quizzes taken by student u until week w as defined in Eq. 2, and:

- $S^{gt(t_w)} = \{o_j \in \mathbb{O} | (o_j, t_j) \in \mathbb{S}_c \cap type(o_j) = \text{quiz} \cap t_j > t_w\}$ is the set of quizzes to complete after week w.

Content Alignment is defined as the number of videos watched by the student until week w, divided by the total number of videos scheduled for the period of consideration. Good-performing students are expected to complete all videos for the period of consideration (value close to 1), while low-performing students may complete less videos than those proposed (value close to 0). Given a student u and the week w of the course, this feature is computed as:

$$\frac{|V_u \cap S^{leq(t_w)}|}{|S^{leq(t_w)}|} \tag{5}$$

where:

- $V_u = \{o_j | i_j = (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \cap \text{type}(o_j) = video\}$ is the set of videos watched by student u until week w.

- $S^{leq(t_w)} = \{o_j \in \mathbb{O} | (o_j, t_j) \in \mathbb{S}_c \cap type(o_j) = \text{video} \cap t_j \leq t_w\}$ is the set of videos to watch by week w.

Content Anticipation is defined as the number of videos completed by the student among those in subsequent weeks of the current week of study. For instance, if a video is due the next week, we expect that good-performing students might watch them earlier, anticipating the deadline stated in the platform (value close to 1). On the other hand, low-performing students may tend to delay the completion of videos (value close to 0). Given a student u and the week w of the course, this feature is computed as:

$$\frac{|V_u \cap S^{gt(t_w)}|}{|S^{gt(t_w)}|} \tag{6}$$

where V_u is the set of videos watched by student u until week w as defined in Eq. 5, and:

- $S^{gt(t_w)} = \{o_j \in \mathbb{O} | (o_j, t_j) \in \mathbb{S}_c \cap type(o_j) = \text{video} \cap t_j > t_w\}$ is the set of videos to watch after week w.

Student Shape is defined as the student's tendency of receiving the maximum grade in a quiz at the first attempt in a row. Good-performing students are expected to consecutively receive the maximum grade in quizzes at the first attempt (value close to 1); students experiencing difficulties may require multiple attempts on each quiz, before getting the maximum grade (value close to 0). Given a student u and the week w of the course, this feature is computed as:

$$\frac{1}{\sum_{(p_i,l_i)\in\mathbb{P}} p_i} \sum_{(p_i,l_i)\in\mathbb{P}} \frac{p_i \cdot l_i}{|\{p_i | (p_i, l_i) \in \mathbb{P} \ \cap \ l_i = 1\}| + \epsilon} \tag{7}$$

where $\mathbb{P} = \{(p_0, l_0), \ldots, (p_n, l_n)\}$ represents a series counting how many quizzes the student consecutively receives the maximum grade ($l_i = 1$) or failed ($l_i = 0$) at the first attempt in a row. For instance, if a student gets the maximum grade for the first five quizzes at the first attempt in a row, then is wrong in two quizzes at the first attempt, and then receives the maximum grade for ten quizzes at the first attempt in a row, $\mathbb{P}$ would be equal to $\{(5, 1), (2, 0), (10, 1)\}$.

Student Speed is defined as the average time passed between two consecutive attempts for the same quiz, among those taken by the student. This feature captures intrinsic behavior of students who take the quiz, spending less time or more time to attempt it, on average. Given a student u and the week w of the course, this feature is computed as:

$$\frac{1}{|Q_u|} \sum_{q \in Q_u} \sum_{i=1}^{|t_q|} \frac{|t_q^i - t_q^{i-1}|}{|t_q|} \tag{8}$$

where Q_u is the set of quizzes taken by student u until week w as defined in Eq. 2, and:

- $t_q = [t_j | (t_j, a_j, o_j, d_j) \in \mathbb{I}_u \ \cap \ o_j = q \ \cap \ t_j > t_{j-1})]$ are timings between trials for u on q, chronologically.

In the rest of the paper, we will refer to our set by *Ours*.

4.2 Experimental Evaluation

In this section, we benchmark our new feature set against the eight feature sets presented in prior work (see Section 3), on early success prediction in flipped courses. For convenience, we will use author-based labels to identify feature sets throughout the paper, but we will be more interested in contrasting the impact of features in those papers based on what they implicitly measure (not based on the authors).

4.2.1 Experimental Setup

Protocol. For each dataset, we applied a train-test evaluation, i.e. parameters were fit on the training data set and the performance of the models was evaluated on the test data set. We performed all experiments using Random Forest (RF) classifiers, known to achieve a good trade-off between prediction accuracy and interpretability. Performance of all models was computed using a **nested** student-stratified (i.e. dividing the folds by students) 10-fold cross validation. The same folds were used for all experiments, across feature sets. We optimized the hyper-parameters of RFs via Grid Search in Scikit-Learn. Specifically, we tuned the following hyper-parameters: number of estimators $(25, 50, 100, 200, 300, 500)$, the maximum number of features $(sqrt, None, log2)$, and the splitting criterion $(gini, entropy)$. More extensive grids were run, but they did not show any substantial improvement. To be precise, we determined the set of optimal hyper-parameters as follows: within each iteration, we ran an inner student-stratified 10-fold cross-validation on the training set in that iteration, and selected the combination of hyper-parameter values yielding the highest accuracy on the inner cross-validation. Note that we trained RFs by weeks: the RF for week w of a given course was trained on data collected up to week w. To obtain the input features for RF for week w, we computed the weekly features for the selected feature set and averaged them.

Data Set: LA-Flip. We consider a Linear Algebra course for undergraduate students taught in a flipped format for 10 weeks at EPFL. Typical pre-class work included a list of video lectures and online quizzes from a Linear Algebra MOOC. The final exam grade, lying between 0 and 6, with 4 as passing threshold, is considered as a measure for students' performance. The repeating students were filtered out, given that their repeated exposure to the material might add a bias to our findings. The final dataset consists of clickstream data from 214 students, with 41% of them failing the course. The study was approved by the university's ethics committee (HREC No. 058-2020/10.09.2020).

4.2.2 Observations

We evaluated the predictive accuracy of RF classifiers trained on the different feature sets extracted from LA-Flip under a binary classification that aims to identify passing and failing students early, as described in Section 2. We further also trained RF classifiers only on the most important features selected from all features (denoted as *EnsembleAll*) and from all features except ours (denoted as *EnsembleButOurs*). Figure 2 reports the balanced accuracy, the area under the ROC curve (AUC), and the individual percentage of passing and failing students correctly identified (recall) for each feature set over all weeks and folds.

The lowest-performing feature sets appear those monitoring students' regularity (orange) and attendance and utilization rates (blue). Hence, a first conclusion we can draw is:

Highlight #1. *Regularity and attendance/utilization features, powerful in MOOCs, do not allow to distinguish passing from failing students in the considered flipped course.*

The feature sets mostly related to video-clicking behavior, such as those from Lemay & Doleck, do not lead to substan-

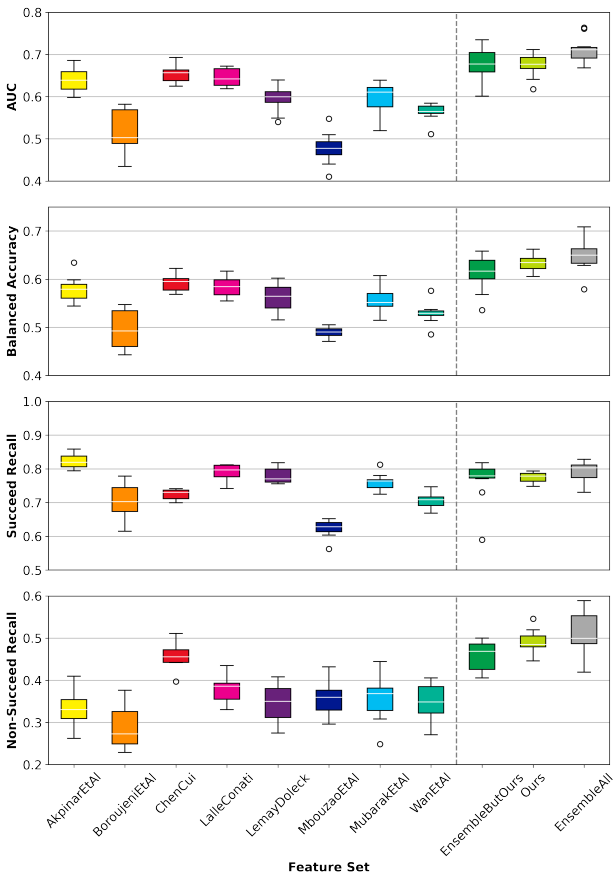


Figure 2: [LA-Flip]. Effectiveness of a RF classifier trained on separate feature sets and on ensembles. Our feature set is essential to increase the effectiveness of the classifier, especially in terms of Non-Succeed (failing students) Recall.

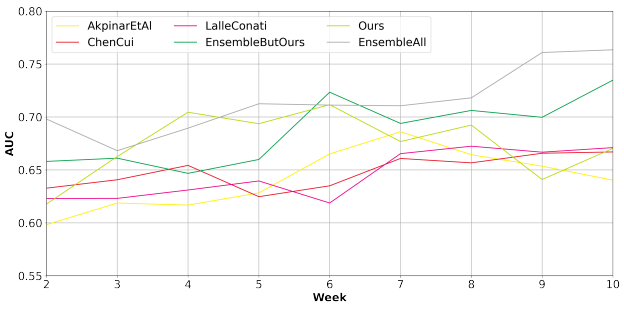


Figure 3: [LA-Flip]. AUC for the best six feature sets. The ensemble of all features (grey) leads to an increase in effectiveness, with respect to considering feature sets separately.

tial differences from each other and all achieved a balanced accuracy between 55% and 59% (similarly for AUC). This finding might reveal an intrinsic limit for video features in predicting student success from pre-class activities. Our results also raise the question on how and why a certain type of video features should be preferred compared to others.

Highlight #2. *In this flipped course, there are minimal differences in performance among video-usage features; an intrinsic predictive limit for video-usage features exists.*

This motivates investigation on the impact of features targeting quiz usage. In this direction, the features proposed by Wan et al. cover a range of raw counting and timing measures that target quizzes. Figure 2 shows that this feature set is even worse that just using video features. Conversely, by measuring more complex patterns in quiz consumption, our feature set led to a balanced accuracy of 67% (similarly for AUC). To identify the aspect our features make the difference at, we considered the percentage of passing and failing students correctly classified, as shown in the two bottom plots in Figure 2. While there are no substantial differences among our feature set and the other ones in identifying passing students (Succeed Recall), a clear improvement is obtained in the detection of failing students (Non-Succeed Recall), fundamental to ensure fewer students are left behind. The impact of our features can be also appreciated across weeks in Figure 3. Our features allowed the ensemble to be effective in the first weeks, while both ours and other features jointly led to an improvement in the second part of the course. Given our results and the characteristics of our features, we can observe that:

Highlight #3. *Extracting fine-grained features that model alignment, anticipation and strength of video/quiz usage results in higher predictive power on failing students.*

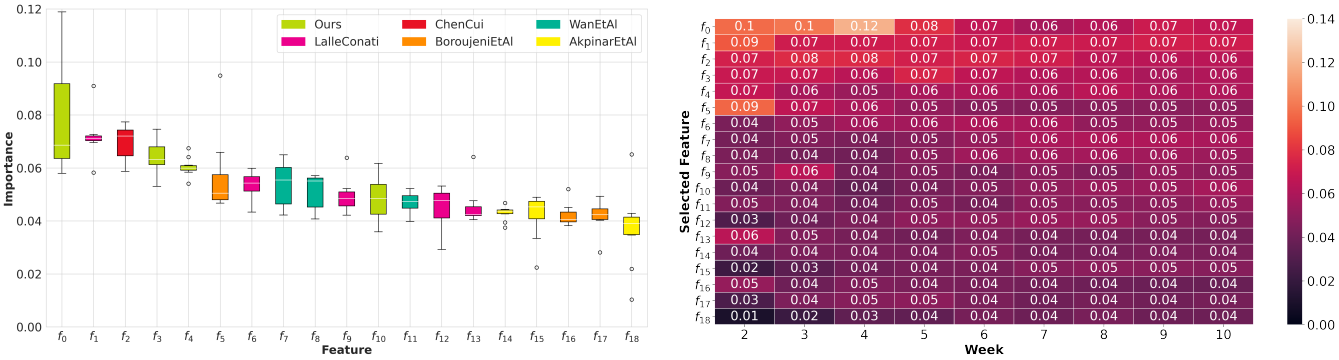
Though considering the feature sets separately allowed us to perform a fine-grained assessment and have an estimation of their predictive power, it remains unclear how the effectiveness of early predictors can be improved by training models with an ensemble of all features and to what extent the importance of the considered features varies. Hence, on the right side of the plots in Figure 2, we present the results achieved by a RF classifier only with the most important features selected from all features and from all features except ours. It can be observed that the optimal ensemble of features without ours results in lower performance, compared to the optimal ensemble that uses also our features. The optimal ensemble of all feature has an AUC score consistently higher than 0.70. To inspect what drives success prediction, we computed the feature importance over weeks and folds, and reported in Figure 4 the importance of features (short description in Table 1) selected by RF. Looking at importance scores in Figure 4(a), we observe that:

Highlight #4. *The extent to which students anticipate content consumption, the tendency of learning during weekends, the proportion of watched videos, and the strength of their performance in quizzes, had the highest importance.*

Figure 4(b) shows that the difference in importance across features is more evident in the first weeks. This finding emphasizes the fact that selecting appropriate features is more crucial when interested in very early predictions.

5. EARLY PREDICTORS OVER COURSES

Our exploratory analysis revealed interesting patterns on the predictive power and importance of a range of features. However, it remains under-explored the extent to which the patterns identified in that flipped course hold also in courses with other structures and educational settings. To this end,



(a) Average feature importance across weeks.

(b) Feature importance over weeks.

Figure 4: [LA-Flip]. Importance of the best nineteen features selected by a RF classifier from the ensemble of all feature sets. Four features of our set have been selected as important. Table 1 lists the Feature IDs and the short description of each feature.

Table 1: [LA-Flip]. Description of the most important nineteen features selected by a RF classifier from the ensemble of all feature sets, showed in decreasing order of importance. Four features of our set have been selected among the top eleven.

ID	Set	Name	Short Description
f_0	Ours	CompetencyAnticipation	The extent to which the student approaches soon a quiz provided in subsequent weeks.
f_1	LalleConati	WeeklyPropWatched-Avg	The proportion of videos the student watched, counting repeating segments.
f_2	ChenCui	RatioClicksWeekendDay	The ratio between clicks happened during weekend and weekdays.
f_3	Ours	ContentAnticipation	The extent to which the student approaches soon a video provided in subsequent weeks.
f_4	Ours	CompetencyStrength	The extent to which a student passes a quiz getting the maximum grade with a low number of trials.
f_5	BoroujeniEtAl	RegWeeklySim-M2	The extent to which the student has a similar distribution of workload among weekdays across weeks.
f_6	LalleConati	WeeklyPropInter-Std	The standard deviation of the time the student spent while interrupting a video, across videos.
f_7	WanEtAl	NumSubmissionCor	The average number of quizzes attempted and correct.
f_8	WanEtAl	NumSubmissions-Avg	The number of submissions required to pass a quiz, on average.
f_9	LalleConati	WeeklyPropInter-Avg	The average time the student spent while interrupting a video, across videos.
f_{10}	Ours	StudentShape	The extent to which the student receives the maximum grade in quizzes at the first attempt in a row.
f_{11}	WanEtAl	NumSubmissionPerCorrect	Percentage of the correct quiz submissions with respect to the total submissions.
f_{12}	LalleConati	WeeklyPropReplayed-Avg	The proportion of videos the student re-watched, not counting repeating segments.
f_{13}	LalleConati	FrequencyEvent-VideoPlay	The frequency of the video play action in the students' online sessions.
f_{14}	AkpinarEtAl	QCheck-QCheck-VLoad	The amount of times the student checks twice a given quiz and then go to load a video.
f_{15}	AkpinarEtAl	VPlay-VPause-VLoad	The amount of times the student plays a video, pause and then load the next one.
f_{16}	BoroujeniEtAl	RegPeriodicity-M3	The extent to which the daily study pattern is repeating over weeks (e.g., same days of the week).
f_{17}	BoroujeniEtAl	RegWeeklySim-M1	The extent to which the student works on the same weekdays.
f_{18}	AkpinarEtAl	VStop-PCheck-VLoad	The amount of times the student stops a video, attempts a quiz and then load the next video.

we extended our analysis to a flipped course in a different domain (Functional Programming, only video data in pre-class activities), a MOOC in the same domain (Linear Algebra, both videos and quizzes), and a MOOC from a different domain (Functional Programming, only video interactions).

5.1 Experimental Setup

Protocol. In this experiment, we followed the steps described in Section 4.2.1, with few exceptions. Specifically, for each data set, we considered only classifiers trained with the optimal ensemble of all features proposed in prior work plus the ones proposed in this paper. To obtain the input features for the RF classifier on week w, we computed the weekly features for all feature sets; then averaged features of the same week, and finally averaged across weeks till week w. For each course, we computed the most important features from the ensemble (eight existing sets and ours) based on the average importance of the features across folds and weeks. The study was approved by the university's ethics committee (HREC No. 096-2020/09.04.2020).

Data Set: FP-Flip. We consider one stream of a Functional Programming course taught to EPFL Master's students in a flipped manner for 10 weeks. The preparatory work included a list of videos from a Functional Programming MOOC. Repeating students were filtered out. Being a Master's course with a failing percentage of only 5%, we considered whether a student's final course grade (lying between 0 and 6) was above the average grade over all students as a success label. The dataset consists of clickstreams from 218 students, with 38% of them being below average.

Data Set: LA-MOOC. The content used in pre-class activities within LA-Flip was also provided by EPFL instructors on an external MOOC platform in form of three separate MOOCs, with the first MOOC being equivalent to the first 4 weeks of the flipped course, the second MOOC equivalent to week 5 to week 8, and the third MOOC equivalent to the last 3 weeks. Given that the first 4 weeks of LA-Flip were delivered in a traditional manner, we excluded the first MOOC from our study. We also excluded the third MOOC, given that the number of enrolled students was barely small. To sum up, our study in this paper considers only the second MOOC that covers the second part (weeks 5 to 8) of the flipped course. To pass the course, it is mostly necessary to obtain at least 60% of the total points for each assignment. Hence, we used this rule as a way to measure success in our study. The final data set consists of clickstream data from 170 students, with 33% of them failing the course.

Table 2: Features selected as important by RF classifiers for the ensemble of features for each course.

Set	Name	Short Description	LA-Flip	FP-Flip	LA-MOOC	FP-MOOC
AkpinarEtAl	QCheck-QCheck-QCheck	The amount of times the student checks three times the same quiz.		✓		
	QCheck-QCheck-VLoad	The amount of times the student checks twice the quiz and then go to load a video.	✓			
	VPlay-VPlay-VPlay	The amount of times the student clicks for three consecutive times on play for three different videos.			✓	
	VPlay-VPause-VLoad	The amount of times the student plays a video, pause and then load another one.	✓			
	VPlay-QCheck-QCheck	The amount of times the student plays a video, then checks twice a quiz.				✓
	VPlay-VStop-VPlay	The amount of times the student plays a video, stops, and plays another one.		✓		
	VPause-VSpeedChange-VPlay	The amount of times the student pauses a video, changes the speed, and re-plays it.		✓		
	VStop-VPlay-VSeek	The amount of times the student stops a video, then re-play it and seek to a given part.		✓		
	VStop-VCheck-VLoad	The amount of times the student stops a video, checks a quiz and then load another video.	✓			
BoroujeniEtAl	DelayLecture	The average delay in viewing video lectures, as soon as they are released.		✓	✓	✓
	RegWeeklySim-M1	The extent to which the student works on the same weekdays across weeks.	✓	✓		
	RegWeeklySim-M2	The extent to which a student has a similar distribution of workload among weekdays across weeks.	✓		✓	✓
	RegWeeklySim-M3	The extent to which the time spent on each day of the week is similar for different weeks of the course.		✓		
	RegPeriodicity-M1	The extent to which the hourly pattern of student's activities is repeating over days.				✓
	RegPeriodicity-M3	If the daily study pattern is repeating over weeks (e.g. is active on Monday and Tuesday in every week).	✓			✓
	RegPeakTime-M1	The extent to which students' activities are centered around a particular hour of the day.				✓
	RegPeakTime-M2	The extent to which students' activities are centered around a particular day of the week.				✓
ChenCui	RatioClicksWeekendWeekdays	The ratio between clicks in weekdays and weekends.	✓	✓	✓	✓
	TimeSession-Avg	The average amount of time spent from a login to the end of the session.			✓	
	TimeSession-Std	The standard deviation of time spent from a login to the end of the session.				✓
	TimeBetweenSessions	The average amount of time passed between two sessions for a student.				✓
	TotalClicks-Weekdays	The number of clicks performed by a student over weekdays.			✓	
LalleConati	PauseDuration-Avg	The average amount of time spent in pause while interacting with a video.		✓		
	SeekLength-Std	The extent to which the seek length varies across videos.		✓		
	PauseDuration-Std	The extent to which the pause duration varies across videos.		✓		
	TimeSpeedingUp-Avg	The average amount of time spent with higher than 1x speed while playing a video.		✓		
	TimeSpeedingUp-Std	The extent to which the time spent speeding up higher than 1x the videos varies.		✓		
	WeeklyPropWatched-Avg	The proportion of videos the student watched, counting repeating segments.	✓			
	WeeklyPropInter-Avg	The average time the student spent in interrupting a video.	✓			
	WeeklyPropInter-Std	The deviation of the time the student spent in interrupting a video.	✓			
	WeeklyPropReplayed-Avg	The proportion of videos the student re-watched, counting repeating segments.	✓			
	WeeklyPropReplayed-Std	The deviation of the proportion of videos the student re-watched, counting repeating segments.			✓	
	FrequencyEvent-VideoPlay	The frequency of the play event in the students' sessions.	✓			✓
MubarakEtAl	SpeedPlayBack-mean	The average speed the student used to play back a video.			✓	✓
WanEtAl	NumSubmissionsCor	The number of quizzes attempted and correct.	✓			
	NumSubmissions-Avg	The number of submissions performed for a quiz, on average.	✓			✓
	NumSubmissionPerCorrect	The percentage of the correct quiz submissions with respect to the total submissions.	✓		✓	
	NumSubmissionDistinct	The total number of distinct problems attempted by the student.			✓	
Ours	CompetencyAnticipation	The extent to which the student approaches soon a quiz provided in subsequent weeks.	✓			
	ContentAnticipation	The extent to which the student approaches soon a video provided in subsequent weeks.	✓			
	CompetencyStrength	The extent to which a student passes a quiz getting the maximum grade with a low number of trials.	✓		✓	
	StudentShape	The extent to which the student receives the maximum grade in quizzes at the first attempt in a row.	✓		✓	
	Student Speed	The average amount of time passed between two submissions for the attempted quizzes.			✓	

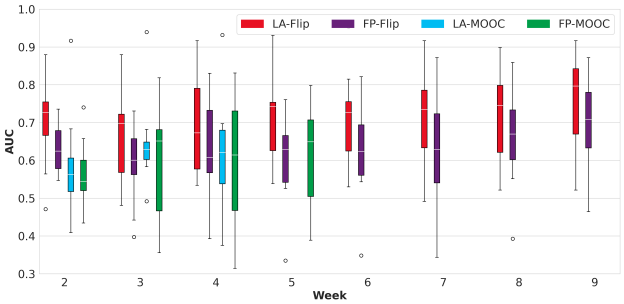


Figure 5: AUC scores per week for RF classifiers trained on feature ensembles. Flipped courses (*-Flip) last 10 weeks; LA-MOOC (FP-MOOC) last 4 (6) weeks.

Data Set: FP-MOOC. The content delivered in pre-class activities in FP-Flip was also provided by EPFL instructors on an external MOOC platform in form of two separate MOOCs, with the first MOOC being equivalent to the first 6 weeks of the flipped course and the second MOOC to the subsequent weeks. No data was available on the second MOOC, so we limited our study to only the first MOOC (week 1 to 6 of the flipped period of FP-Flip). To pass this MOOC, 80% of the total points for each of the five graded assignments are mostly needed. Hence, we used this rule to measure success in our study. The dataset consists of clickstreams from 3,565 students, with 52% failing the course.

5.2 Observations

We evaluated the predictive performance of a RF classifier across weeks for each course for the best ensemble feature set for that course. Figure 5 illustrates the predictive performance across weeks for all four courses. Considering the same course across different settings (flipped or MOOC), it can be observed that RFs trained on flipped course data achieved higher AUC scores than their MOOC counterpart. This difference can be due to multiple reasons, for example the different educational setting or the way the passing rule for the course is set up. Considering courses in the same setting (LA-Flip VS FP-Flip or LA-MOOC VS FP-MOOC), the results show that including quizzes in the LA-Flip course allows to increase the predictive power of the considered classifiers, compared to FP-Flip, that has no quizzes. This can be associated to the fact that quizzes are a good source of information for grasping the students' performance. The same observation is, however, less strong on the MOOC counterpart of the same two courses, highlighting again the high dependency from the educational setting.

In a second part of this experiment, we analyzed the average importance across weeks of the features selected by RFs across courses. Table 2 shows for each feature set and course, whether a given feature has been selected by the corresponding RF classifier. It should be noted that this table includes only features picked at least by a RF classifier across courses. In general, we show that while there is some overlap between the optimal features across courses, the importance of the features highly depends on the setting and structure of the course. The ratio of clicks between weekends and weekdays (ChenCui - RatioClicksWeekendWeekdays) is selected by all classifiers in all settings. Other features with a good level of generalizability are represented by those measuring regularity (BoroujeniEtAl). The other features were picked according to the setting or the structure of the course. In particular, RFs trained on LA-Flip and LA-MOOC assigned a higher importance to features that measure behavior in quizzes (e.g., Ours or WanEtAl). Hence, we can conclude that when available, features on quizzes are frequently selected, regardless of the setting. For courses with no quizzes, namely FP-Flip and FP-MOOC, the predictive power of RFs

is mainly based on regularity and fine-grained video usage (e.g., features on time spent in a video, e.g., LalleConati).

Highlight #5. *When quizzes are included in the schedule, quiz-related features are frequently selected as important. This is stronger in flipped than MOOC settings. When only videos are available, the predictive power mainly derives from regularity and fine-grained video-related features.*

For the same course in different settings, namely LA-Flip VS LA-MOOC and FP-Flip VS FP-MOOC, the optimal feature set heavily changed. In LA courses, quiz-related features were more important in the flipped context, while session-based features were more important in MOOCs (e.g., those from ChenCui). The latter finding holds for FP courses as well. Specifically, RFs trained on the MOOC version consistently selected features related to the students' session. Another observation for FP is that in the flipped version, tri-grams (ApkinarEtAl) and fine-grained video usage features (LalleConati) were picked; in the MOOC, regularity and session-based features were more important. To sum up, according to Table 2:

Highlight #6. *Predictors in flipped settings often rely on features based on tri-grams and fine-grained video consumption. Conversely, predictors in MOOCs consider regularity and session-based features as important. Quiz-related feature are picked in both settings, when quizzes are available.*

6. DISCUSSION

In this section, we connect the main findings coming from the individual experiments and present the implications and limitations of our study in the early success prediction task.

Course-Related Observations. A challenge, as our work shows, lies on the generalizability of feature predictive power across courses. The variability of the results when repeating the exact same experiment with data from different courses (or slightly different settings) is very high. It is therefore challenging to understand when, why, and how a feature tested on a given course could be re-used for other courses.

Highlight #7. *The predictive power of features does not often generalize across courses with different structures and educational settings. This observation is stronger with respect to the courses structure than between flipped and MOOC settings.*

This observation affects the scalability of early predictors. Being so course-dependent, identifying and enabling features predictive of student success for a given course can take hours or days, given that the intellectual and experimental work needs to be replicated on courses, case by case.

Highlight #8. *The lack of feature predictive power generalizability questions the extent to which a feature can be scaled across courses with the same structure/setting.*

Our experiments also showed that including quizzes in pre-class activities leads to substantial improvements in effectiveness. Hence, success prediction is driven by complex relationships between students' characteristics and the course domain, structure, and educational setting.

Data-Related Observations. Research in the area of early success prediction is often conducted on data extracted from online activities only. Even in our case study (for LA-flip), we could not rely on data collected in class, missing an important segment of learning. Moreover, clickstreams in this study do not cover other relevant interactions such as those in forums. In flipped courses, most (non-digitalized) discussions happen in class, and the forum is mainly used by teachers for announcements.

Highlight #9. *Early success prediction in flipped courses would benefit from including data coming from offline activities (e.g., in class).*

Workflow-Related Observations. To establish reproducibility, the description of the proposed features should go beyond plain-text only. Our formulation in this paper can be re-used to define features as formulas, making it easier to replicate them, especially when no source code is provided.

Highlight #10. *Feature descriptions can be accompanied by their mathematical formulation to ease reproducibility. When possible, sharing the code can facilitate their re-use.*

Though we validated the current features on RFs, other classifiers were not presented. However, RFs often provide the best trade-off between effectiveness and interpretability (the latter was fundamental for our study) and our framework makes it easy to run this analysis on other classifiers. Given that other classifiers (e.g., Support Vector Machines) gave worse (or comparable) results in the preliminary experiments we ran, our results depict a valid picture of feature predictive power.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we analyzed recent features for early success prediction in flipped and online courses. First, we investigated the predictive power of eight existing feature sets and a novel feature set proposed in this paper on a flipped course. We benchmarked the predictive power of features using a RF classifier, and discussed the ensemble feature set optimal for that course. We then extended our analysis to courses with other settings (MOOCs), domains, and structures, showing that the optimal ensemble and its predictive power vary. Our work calls for generalizable early predictors across courses with different characteristics. To promote research in this field, we also publicly release the source code developed during our study (see the footnote in Section 1).

In future work, we plan to extend our analysis to other features (e.g., based on in-class data), and types of student success tasks (e.g., grade prediction). We also plan to analyze more advanced classifiers and to devise robust classifiers across courses before testing them in the real world.

Acknowledgments We thank Himanshu Verma (TU Delft, formerly at EPFL) and Patrick Jermann and Francisco Pinto (EPFL Center for Digital Education) for the valuable input and support on the data sharing and manipulation.

8. REFERENCES

[1] N. Akpinar, A. Ramdas, and U. Acar. Analyzing student strategies in blended courses using clickstream data. In *Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020*, pages 6–17. Int. Educat. Data Mining Society, 2020.

[2] B. J. Beatty, Z. Merchant, and M. Albert. Analysis of student use of video in a flipped classroom. *TechTrends*, 63(4):376–385, 2019.

[3] M. S. Boroujeni, K. Sharma, L. Kidzinski, L. Lucignano, and P. Dillenbourg. How to quantify student's regularity? In *Proceedings of the 11th European Conference on Technology Enhanced Learning, EC-TEL 2016*, volume 9891 of *Lecture Notes in Computer Science*, pages 277–291. Springer, 2016.

[4] F. Chen and Y. Cui. Utilizing student time series behaviour in learning management systems for early prediction of course performance. *Journal of Learning Analytics*, 7(2):1–17, 2020.

[5] J. Gardner and C. Brooks. Student success prediction in moocs. *User Modeling and User-Adapted Interaction*, 28(2):127–203, 2018.

[6] N. Goedhart, N. Blignaut-van Westrhenen, C. Moser, and M. Zweekhorst. The flipped classroom: supporting a diverse group of students in their learning. *Learning Environments Research*, 22(2):297–310, 2019.

[7] H. He, Q. Zheng, B. Dong, and H. Yu. Measuring student's utilization of video resources and its effect on academic performance. In *Proceedings of the 18th IEEE International Conference on Advanced Learning Technologies, ICALT 2018*, pages 196–198. IEEE Computer Society, 2018.

[8] J. Jovanović, D. Gašević, S. Dawson, A. Pardo, N. Mirriahi, et al. Learning analytics to unveil learning strategies in a flipped classroom. *The Internet and Higher Education*, 33(4):74–85, 2017.

[9] J. Jovanovic, N. Mirriahi, D. Gasevic, S. Dawson, and A. Pardo. Predictive power of regularity of pre-class activities in a flipped classroom. *Computers & Education*, 134:156–168, 2019.

[10] C. Lai and G. Hwang. A self-regulated flipped classroom approach to improving students' learning performance in a mathematics course. *Computers & Education*, 100:126–140, 2016.

[11] S. Lallé and C. Conati. A data-driven student model to provide adaptive support during video watching across moocs. In *Proceedings of the 21st International Conference on Artificial Intelligence in Education, AIED 2020*, volume 12163 of *Lecture Notes in Computer Science*, pages 282–295. Springer, 2020.

[12] J. Lee and H. Choi. Rethinking the flipped learning pre-class: Its influence on the success of flipped learning and related factors. *British Journal of Educational Technology*, 50(2):934–945, 2019.

[13] D. J. Lemay and T. Doleck. Grade prediction of weekly assignments in MOOCS: mining video-viewing behavior. *Education and Information Technologies*, 25(2):1333–1342, 2020.

[14] G. S. Mason, T. R. Shuman, and K. E. Cook. Comparing the effectiveness of an inverted classroom to a traditional classroom in an upper-division engineering course. *IEEE Transactions on Education*, 56(4):430–435, 2013.

[15] B. Mbouzao, M. C. Desmarais, and I. Shrier. Early prediction of success in MOOC from video interaction features. In *Proceedings of the 21st International Conference on Artificial Intelligence in Education, AIED 2020*, volume 12164 of *Lecture Notes in Computer Science*, pages 191–196. Springer, 2020.

[16] A. A. Mubarak, H. Cao, and S. A. M. Ahmed. Predictive learning analytics using deep learning model in moocs' courses videos. *Education and Information Technologies*, 26(1):371–392, 2021.

[17] E. M. W. Ng. Integrating self-regulation principles with flipped classroom pedagogy for first year university students. *Computers & Education*, 126:65–74, 2018.

[18] J. O'Flaherty and C. Phillips. The use of flipped classrooms in higher education: A scoping review. *The Internet and Higher Education*, 25:85–95, 2015.

[19] S. Park and N. H. Kim. University students' self-regulation, engagement and performance in flipped learning. *European Journal of Training and Development*, 2021.

[20] W. W. Porter, C. R. Graham, K. A. Spring, and K. R. Welch. Blended learning in higher education: Institutional adoption and implementation. *Computers & Education*, 75:185–195, 2014.

[21] A. A. Rahman, B. Aris, M. S. Rosli, H. Mohamed, Z. Abdullah, and N. Mohd Zaid. Significance of preparedness in flipped classroom. *Advanced Science Letters*, 21(10):3388–3390, 2015.

[22] M. Shih, J. Liang, and C. Tsai. Exploring the role of university students' online self-regulated learning in the flipped classroom: a structural equation model. *Interact. Learn. Environ.*, 27(8):1192–1206, 2019.

[23] N. A. Uzir, D. Gasevic, W. Matcha, J. Jovanovic, and A. Pardo. Analytics of time management strategies in a flipped classroom. *Journal of Computer Assisted Learning*, 36(1):70–88, 2020.

[24] J. N. Walsh and A. Risquez. Using cluster analysis to explore the engagement with a flipped classroom of native and non-native english-speaking management students. *The International Journal of Management Education*, 18(2):100381, 2020.

[25] H. Wan, J. Ding, X. Gao, and K. Liu. Supporting quality teaching using educational data mining based on openedx platform. In *Proceedings of the IEEE Frontiers in Education Conference, FIE 2017*, pages 1–7. IEEE Computer Society, 2017.

[26] H. Wan, K. Liu, Q. Yu, and X. Gao. Pedagogical intervention practices: Improving learning engagement based on early prediction. *IEEE Transactions on Learning Technologies*, 12(2):278–289, 2019.

[27] K. Wang and C. Zhu. Mooc-based flipped learning in higher education: students' participation, experience and learning performance. *International Journal of Educational Technology in Higher Education*, 16(1):1–18, 2019.

[28] R. M. Yilmaz and O. Baydas. An examination of undergraduates' metacognitive strategies in pre-class asynchronous activity in a flipped classroom. *Educational Technology Research and Development*, 65(6):1547–1567, 2017.

Early Prediction of Conceptual Understanding in Interactive Simulations

Jade Cock
EPFL
jade.cock@epfl.ch

Mirko Marras
EPFL
mirko.marras@epfl.ch

Christian Giang
EPFL
christian.giang@epfl.ch

Tanja Käser
EPFL
tanja.kaeser@epfl.ch

ABSTRACT

Interactive simulations allow students to independently explore scientific phenomena and ideally infer the underlying principles through their exploration. Effectively using such environments is challenging for many students and therefore, adaptive guidance has the potential to improve student learning. Providing effective support is, however, also a challenge because it is not clear how effective inquiry in such environments looks like. Previous research in this area has mostly focused on grouping students with similar strategies or identifying learning strategies through sequence mining. In this paper, we investigate features and models for an early prediction of conceptual understanding based on clickstream data of students using an interactive Physics simulation. To this end, we measure students' conceptual understanding through a task they need to solve through their exploration. Then, we propose a novel pipeline to transform clickstream data into predictive features, using latent feature representations and interaction frequency vectors for different components of the environment. Our results on interaction data from 192 undergraduate students show that the proposed approach is able to detect struggling students early on.

Keywords

skip-grams, early classification, interactive simulations, conceptual understanding

1. INTRODUCTION

Over the last years, interactive simulations have been increasingly used for science education (e.g, the PhET simulations alone are used over $45M$ times a year [1]). Interactive simulations allow students to engage in inquiry-based learning: they can design experiments, take measurements, and test their hypotheses. Ideally, students discover the principles and models of the underlying domain through their own exploration [2], but students often struggle to effectively learn in such environments [3, 4, 5]. A possible reason for this is that interactive simulations are usually complex and unstructured environments allowing students to choose their own action path [6]. Providing adaptive guidance to students has therefore the potential to improve learning outcomes.

Implementing effective support in interactive learning environments is a challenge in itself: the complexity of the environment makes it difficult to define a priori how successful student behaviour looks like. Previous research has focused on leveraging sequence mining and clustering techniques to identify the key features of successful interactions. For example, [7] have used an information theoretic sequence mining approach to detect differences in the interaction sequences of students with high and low prior knowledge, while [8] investigated the effects of prior knowledge activation. Other work [9] focused on detecting behaviours leading to the design of a correct causal explanation. [10] identified key factors for successful inquiry: focusing on an unknown component and building contrastive cases. Similarly, [11] found that the identification of the dependent variable and its isolated manipulations lead to a better quantitative understanding of the phenomena at hand. Another technique is to manually categorise students' log data and use the tags as ground truth for a classifier of successful inquiry behaviour [12]. [13] developed a dashboard displaying information about the mined sequences to guide teachers in building their lessons.

More work has focused on analysing and predicting students' strategies in different types of open ended learning environments (OELEs), such as educational games. Prior research in that domain has, for example, investigated students' problem solving behaviour [14], analysed the effect of scaffolding on students' motivation [15], extracted strategic moves from video learning games [16], detected different types of confusion [17], or identified students' exploration strategies [18].

Most of the previous work on OELEs has performed *a posteriori* analyses. However, in order to provide students with support in real-time, we need to be able to detect struggling students early on. Due to the lack of clearly defined student trajectories and underlying skills, building a model of students' learning in OELEs is challenging. A promising approach for early prediction in OELEs is the use of a clustering-classification framework [19]: in the (first) offline step, students are clustered based on their interaction data and the clustering solution is interpreted. The second step is online: students are assigned to clusters in real-time. This

Jade Cock, Mirko Marras, Christian Giang and Tanja Käser "Early Prediction of Conceptual Understanding in Interactive Simulations". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 161-171. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

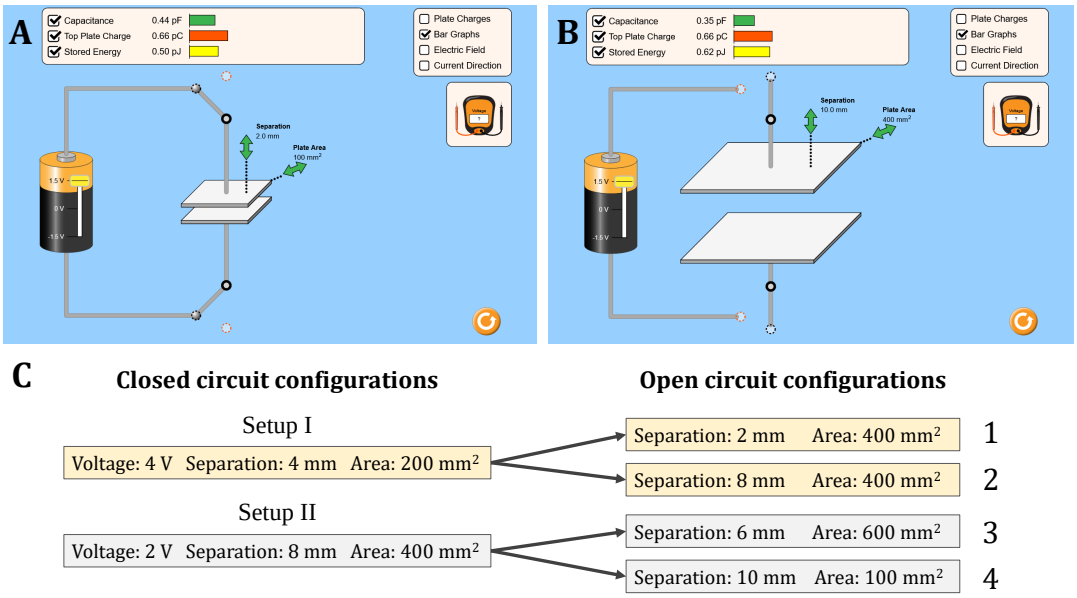


Figure 1: User interface of the PhET Capacitor Lab with different plate parameters for a closed (A) and an open circuit (B). The two initial closed circuit configurations and the resulting four open circuit configurations presented to the participants in the capacitor ranking task (C). (Simulation image by PhET Interactive Simulations, University of Colorado Boulder, licensed under CC-BY 4.0, https://phet.colorado.edu).

framework has been successfully applied to analyse and predict students' trajectories in mathematics learning [20], to differentiate between 'high' and 'low' learners [21], to build student models for interactive simulations [22], or to predict students' exploration strategies in an educational game [23].

In this paper, we aim at early predicting conceptual understanding based on students' log data from an interactive Physics simulation. All our analyses are based on data collected from 192 undergraduate Physics students interacting with a PhET simulation. We propose a novel pipeline for transforming clickstream data into predictive features using latent feature representations and frequency vectors. Then, we extensively evaluate and compare various combinations of predictive algorithms and features on different classification tasks. In contrast to previous work using unsupervised clustering to obtain student profiles [21, 20, 23, 22], our learning activity with the simulation includes a task specifically designed to assess students' conceptual understanding. With our analyses, we address three research questions: 1) Can students' interaction with the data be associated with the gained conceptual understanding? 2) Can conceptual understanding be inferred through sequence mining methods with embeddings? 3) Can the proposed methods be used for early predicting students' conceptual understanding based on partial sequences of interaction data?

Our results show that all tested models are able to predict students' conceptual knowledge with a high AUC when observing students' full sequences (offline). The best models are also able to detect struggling students early on and to provide a more fine-grained prediction of students' conceptual knowledge later during interaction.

2. CONTEXT AND DATA

All experiments and evaluations of this paper were conducted using data from students exploring an interactive simulation. In the following, we describe the learning activity, the data collection, and the categorisation of students' conceptual understanding at the end of the learning activity.

Learning Activity. The data for this work was collected in a user study where participants were asked to engage in an inquiry-based learning activity with the PhET Capacitor Lab simulation[1]. The Capacitor Lab is an interactive simulation with a simple and intuitive interface allowing users to explore the principles behind a plate capacitor (Fig. 1A and B). Specifically, students can load the capacitor by adjusting the battery and observe how the capacitance and the stored energy of the capacitor change when adjusting the voltage, the area of the capacitor plates or the distance between them. After loading the capacitor, the circuit can be opened through a switch, and students can again observe how manipulation of the different components influences capacitance and stored energy. Moreover, the simulation provides a voltmeter, while check boxes in the interface allow users to enable or disable visualisations of specific measures.

Based on this simulation, a learning activity was designed in which participants had to explore the relationships between the different components of the circuit and rank four different capacitor configurations by the amount of stored energy. The configurations were generated based on two initial setups (I and II, respectively) representing capacitors in a *closed circuit* with different settings for battery voltage,

[1]https://phet.colorado.edu/en/simulation/capacitor-lab-basics

plate area and separation (Fig. 1C). For each initial setup, two *open circuit* configurations were generated (i.e. configurations 1 & 2 from setup I and 3 & 4 from setup II) by opening the switch and then changing the values for plate area and separation. To complete this ranking task, participants were allowed to use the simulation for as much time as they needed. It should be noted that the values in the ranking task were chosen outside the ranges of the adjustable values in the simulation such that students could not simply reproduce the four configurations in the simulation, but had to solve the task by figuring out the relationships between the different components and the stored energy.

Data Collection. Data was collected from 214 first-year undergraduate Physics students who completed the capacitor ranking task as part of a homework assignment. While working with the simulation, students' interaction traces (i.e. clicks on check boxes, dragging of components, moving of sliders) were automatically logged by the environment. Moreover, it recorded students' final answers in the ranking task (i.e. the ranking of the four configurations). All data was collected in a completely anonymous way and the study was approved by the responsible institutional review board prior to the data collection (HREC number: 050-2020/05.08.2020). After a first screening of the data, several log files (9) were excluded because of inconsistencies in the data, and another 13 because they had barely any interaction (less than 10 clicks) with the environment. Removing these data points resulted in a data set of 192 students used for our analyses.

Categorisation of conceptual understanding. The design of the ranking task allows to relate students' responses to their conceptual understanding of a capacitor. For this purpose, we analysed the 16 (out of 24 possible) rankings submitted by the students with regards to conceptual understanding and grouped them accordingly. To this end, three concepts of understanding associated with the functioning of capacitors were evaluated in a top-down approach: answers were first separated by those representing an understanding of both the open and closed circuit (label *both*), and those only representing an understanding of the closed circuit (label *closed*). For those answers representing an understanding of both the open and closed circuit, two cases were distinguished. It was assumed that students who chose the only correct ranking of the configurations ("4213") gained an exhaustive understanding of the underlying concepts (label *correct*). Students who instead chose one of the other rankings were assumed to know how plate area and separation influence the stored energy in both the open and closed circuits, but failed to discover the influence of voltage on stored energy (label *areasep*). Within those answers that only represented an understanding of the capacitor's functioning in the closed circuit, we also distinguished between two cases. The first case represents the answer that would be considered correct if the task was to order the four configurations by capacitance instead of energy ("1324", label *capacitance*). Interestingly, 47 students (i.e. 24% of all students) submitted this ranking as an answer. The second case represents all other possible answers (label *other*) that could be submitted if (a part of) the closed circuit was understood.

Based on these three underlying concepts, we generated a decision tree with four leaves (each representing a group with similar conceptual understanding) and mapped all 16 rankings submitted by the students to the leaves (Fig. 2). These generated class labels will serve as ground truth labels for the classification task presented in the following sections.

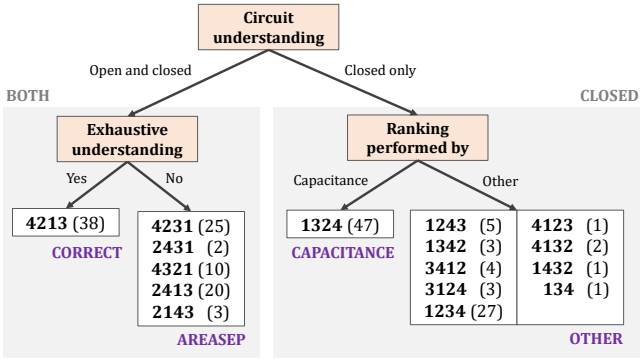


Figure 2: Tree used to map the 16 different rankings submitted by the students to class labels associated with conceptual understanding of a capacitor. The different class labels are indicated in capitalised letters. The numbers in parentheses indicate the number of submissions for each ranking.

3. METHOD

Using our proposed approach, we are interested in predicting the conceptual understanding students gain from interacting with the simulation. Therefore, we are solving a supervised classification problem, i.e. we aim at predicting the class labels (representing students' conceptual understanding) based on the observed student interactions. Our model building process to solve this classification problem consists of four steps (Fig. 3). We first extract the raw clickstream events from the logs and process them into action sequences. We then compute three different types of features for each action sequence and feed them into our classifiers.

Event Logs. From the simulation logs, we extract the clickstream data of each student s as follows: anything between a mouse click/press and a mouse release qualifies as an *event*, while anything between a mouse release and mouse click/press is called *break*. Each *event* is then labelled by the component the user was interacting with at the mouse click, and chronologically arranged with the *breaks* into a sequence.

Action Processing. We distinguish three main components on the platform, whose values can be changed: 1) *the voltage*, 2) *the separation between the plates*, and 3) *the area of the plates*. An action on these components can be conducted in a) *an opened circuit* or b) *a closed circuit*, with the stored energy information display i) *on* or ii) *off*. We categorise each event involving these main actions by the combination of: the action on the component {1), 2), 3)}, the circuit state {a), b)}, and the stored energy display {i), ii)}. Any other event is categorised as 4) *other*.

The sequence of each student s is now composed of chronologically ordered events (divided into the 13 different categories listed above) separated by breaks. The breaks may be caused by the student being inactive due to observing

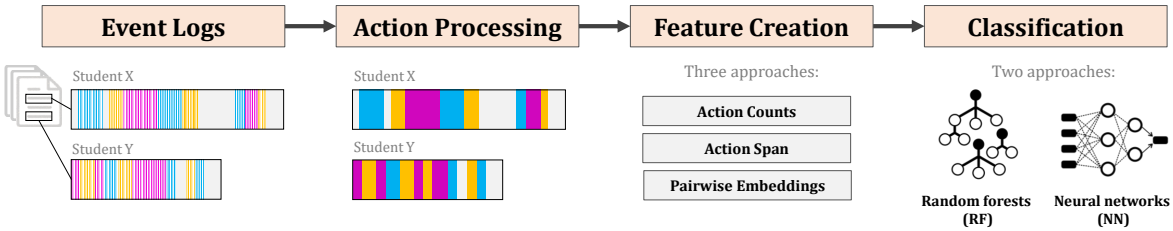


Figure 3: Schematic overview of the classification pipeline. Raw clickstream events are extracted from log data and processed into an action sequence per user. Three types of features are computed for each action sequence. Classification is performed using Random Forests (RF) and Neural Networks (NN).

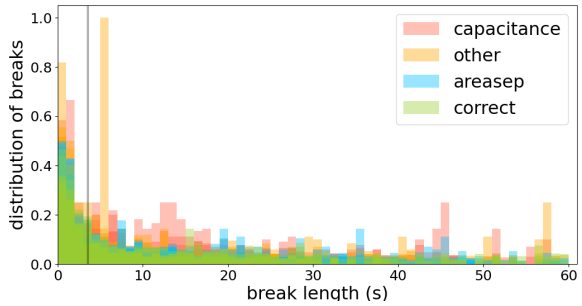


Figure 4: Distribution of break lengths in seconds, across all students. The bold gray line denotes the 60% threshold.

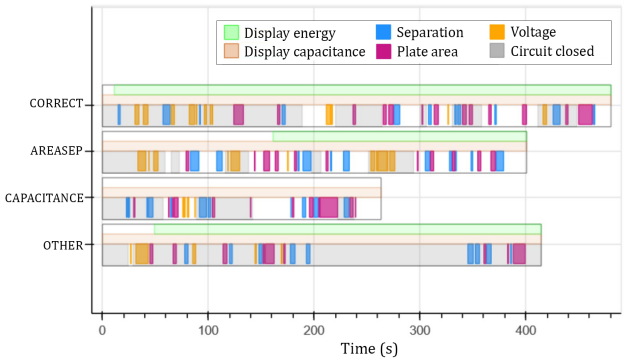


Figure 5: Timeline of an exemplary student for each class label displaying the chronological sequence of interactions with the three main components of the simulation. The green and orange bars indicate whether the student displayed the capacitance and stored energy. The background indicates whether the interactions were conducted in a closed (grey) or open (white) circuit.

the progression of a value, reflecting about an observation, or taking notes. Due to its definition as the period between a mouse release and a mouse click/press, a break may also appear due to logistic reasons, such as moving the mouse from one component in the simulation to another component. Indeed, the students' event sequences consist of many short breaks (Fig. 4). Like stop words in natural texts, our assumption is that these very short breaks, though very frequent in our sequences, do not contain much information. In fact, our classification over students' understanding may be impaired if those noisy states are not removed, like it is the case for sentiment analysis when stop words are not deleted [24]. To determine the threshold at which the breaks are removed, we plot the distribution of inactivity periods, and cut at the elbow of the curve for each student, which corresponds to a delimitation at 60%, i.e. for each student we keep the top 40% of breaks. We then categorise each of our remaining breaks similarly to our main action events: by component 5) *break*, circuit state {a), b)}, and stored energy display {i), ii)}, resulting in four different break categories.

The resulting sequence $\mathbf{r_s}$ for student s is the chronological timeline of the student's events and breaks, divided into 17 categories. We refer to this timeline $\mathbf{r_s}$ as the *raw sequence* of *interactions* for the rest of the paper. We denote the length (corresponding to the total number of interactions of student s) of $\mathbf{r_s}$ with N_s, i.e. $|\mathbf{r_s}| = N_s$. On average, these sequences have a length of $N_s = 67.86 \pm 42.56$. In terms of seconds, the sequences $\mathbf{r_s}$ lasted on average 512.18 ± 435.57 seconds. We also introduce the notion of *time*, which we define related to a student's interactions: at time t the length of the raw sequence of interactions for student s is t, i.e. $|\mathbf{r_{s,t}}| = t$. We denote the maximum time of student s with T_s, corresponding to the full sequence $\mathbf{r_s}$. Figure 5 visualises the timelines for an exemplary student of each class label. It can be observed that for these examples, certain aspects of conceptual understanding could be inferred by visual inspection (e.g. the *capacitance* student never activated the check box to visualise the stored energy). However, other differences in conceptual understanding are more difficult to detect by humans (e.g. the differences between the *correct* and *areasep* students).

Feature Creation. Next, we transform the interactions in each sequence to obtain three different types of features: *Action Counts*, *Action Span*, and *Pairwise Embeddings*.

To obtain the *Action Counts* features $F_{AC,s}$ for a student s, we first transform each interaction within the raw sequence $\mathbf{r_s}$ in a one-hot encoded vector, resulting in a 17-dimensional vector $\mathbf{h_{s,i}}$ for each interaction i and hence, a sequence of

vectors $H_s = \{\mathbf{h_{s,i}}\}$ with $i = 1, ..., N_s$. To compute $\mathbf{f_{AC,s,t}}$ for student s at time step t, we compute the average over $\mathbf{h_{s,i}}$ with $i = 1, ..., t$: $\mathbf{f_{AC,s,t}} = \frac{1}{t}\sum_{i=1}^{t} \mathbf{h_{s,i}}$. By using this aggregating technique, our features are translated to the (averaged) number of times each student has interacted in each of our categories. Therefore, for each student s, we end up with a feature set $F_{AC,s} = \{\mathbf{f_{AC,s,t}}\}$ with $t = 1, ..., T_s$.

The computation of the *Action Span* features $F_{AS,s}$ for a student s is very similar. Rather than looking into the number of times a student s has interacted with each of our components in a particular state, we look into the amount of time (in seconds) s has spent in each of the categories. We first transform every interaction i within $\mathbf{r_s}$ into a 17-dimensional vector $\mathbf{h_{s,i}}$: This vector is 0 for all dimensions but the dimension d corresponding to the category the i_h interaction belongs to. This representation is similar to a one-hot encoded vector, but instead of filling in a 1 at dimension d, we fill in the duration (in seconds) of interaction i. This results in a sequence of vectors $H_s = \{\mathbf{h_{s,i}}\}$ with $i = 1, ..., N_s$. We then compute the feature vector $\mathbf{f_{AS,s,t}}$ for student s at time t in two steps. In a first step, we again average over all vectors $\mathbf{h_{s,i}}$ up to time t, leading to $\tilde{\mathbf{f}}_{\mathbf{AS,s,t}} = \frac{1}{t}\sum_{i=1}^{t} \mathbf{h_{s,i}}$. We then normalise $\tilde{\mathbf{f}}_{\mathbf{AS,s,t}}$ to obtain $\mathbf{f_{AS,s,t}}$. By using this aggregation technique, our feature vector $\mathbf{f_{AS,s,t}}$ represents the relative amount of time student s has spent in each category up to time t. For each student s, we end up with with a feature set $F_{AS,s} = \{\mathbf{f_{AS,s,t}}\}$ with $t = 1, ..., T_s$.

The third feature, *Pairwise Embeddings*, is fundamentally different from the the two other features: we replace each interaction in the raw sequence by an embedding vector which we obtain by training a pairwise skip-gram [25]. The architecture of such a network consists of two dense layers: an embedding layer followed by a classification layer. Usually applied to natural language applications (NLP), its primary goal is to predict the context of a word. Here, our pairwise skip-gram attempts to predict the behavior of a student in the simulation before and after performing a specific interaction. The skip-gram model can be formulated as:

$$\mathbf{p} = \text{softmax}(W_2 \cdot (W_1 \cdot a)) \tag{1}$$

It takes a, an interaction we wish to predict the context of as an input, and outputs $\mathbf{p}$, a probability vector which contains the likelihood of a being surrounded by each possible *interaction*. W_1 and W_2 represent the weight matrices (embeddings). For each interaction a, we feed $2 \cdot w$ pairs into the network, where w is the so-called window size of the model (context). The first element of each pair is a. The second element of each pair, the ground truth label, is one of the w interactions preceding or following a. For example, a window size of $w = 2$ would yield the following pairs for action a: $(a, a_{-2}), (a, a_{-1}), (a, a_{+1}), (a, a_{+2})$.

In our case, to obtain the set of pairs I_s for a student s, we first again transform each interaction within the raw sequence $\mathbf{r_s}$ in a one-hot encoded vector, resulting in a 17-dimensional vector $\mathbf{h_{s,i}}$ for each interaction i and, hence, a sequence of vectors $H_s = \{\mathbf{h_{s,i}}\}$ with $i = 1, ..., N_s$. We then build the input pairs for each interaction i, i.e. $I_{i,s} = \{(\mathbf{h_{s,i}}, \mathbf{h_{s,j}})\}$ with $j \in \{-w, ..., w\} \setminus 0$. We obtain the set of pairs for all students as $I = \{I_s\}$ with $s = 1, .., S$, where S is the total number of students.

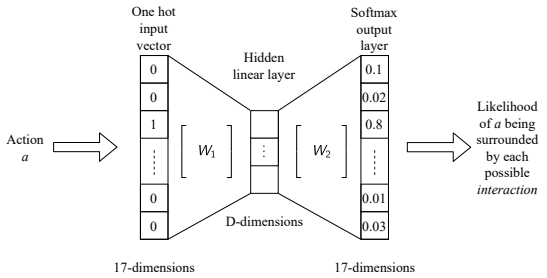


Figure 6: Skip-gram architecture. After training of the skip-gram, the corresponding row of the weight matrix W_1 represents a structure preserving embedding of action a.

After training the skip-gram model on I, we build the features $F_{PW,s}$ for each student s. The weighs W_1 of the hidden layer represent a structure preserving embedding of the interactions. In our case, W_1 has dimensions $17 \times D$ (D is the embedding dimension). For each interaction i of student s, we get the corresponding row r in W_1, i.e. $\mathbf{r_{s,i}} = W_1 \cdot \mathbf{h_{s,i}}$. This again results in a sequence of vectors $R_s = \{\mathbf{r_{s,i}}\}$ with $i = 1, ..., N_s$. To compute $\mathbf{f_{PW,s,t}}$ for student s at time step t, we compute the average over $\mathbf{r_{s,i}}$ with $i = 1, ..., t$: $\mathbf{f_{PW,s,t}} = \frac{1}{t}\sum_{i=1}^{t} \mathbf{r_{s,i}}$. Therefore, for each student s, we end up with a feature set $F_{PW,s} = \{\mathbf{f_{PW,s,t}}\}$ with $t = 1, ..., T_s$.

Classification. To perform the classification task, we explore two different approaches: *Random Forests* and *Fully Connected Deep Neural Networks*.

Random Forests (RFs) are simple, yet powerful machine learning algorithms. They consist of an ensemble of decision trees, each trained on a different subset of samples and a different subset of features. The decisions of each tree are then aggregated to determine the final prediction of a sample. The strength of this method is that overfitting is prevented through the randomisation of training samples and features during the training of each tree and that the strengths of several good classifiers are exploited. While RF classifiers are well tested and efficient to train, they require the input features to have the same dimension for every sample. We therefore train separate RF models for each time step t. The input features for the RF model for time step t are $\{\mathbf{f_{M,s,t}}\}$, with $M \in AC, AS, PW$ and $s = 1, ..., S$, where S denotes the number of students. The output of the RF is a vector $\mathbf{p_{RF,M,s,t}}$ of dimension C (with C denoting the number of classes) for each student s, which represents the probability of each class.

Neural Networks (NNs) were built with the idea of emulating neurons firing in our brain: their nodes are to the neurons what their edges are to their axons. The advantage of those deep networks is that they are able to model non-linear decision boundaries. However, the back propagation calculations make them relatively slow to train. In this work, we use a *Fully Connected Deep Neural Network* consisting of d hidden dense layers and one classification layer with a softmax activation. Similar to RFs, our NN model requires features to have the same dimension for all the samples. We therefore also train the NN models for fixed points in time. The input features for the NN model for time step t are

$\{\mathbf{f_{M,s,t}}\}$, with $M \in AC, AS, PW$ and $s = 1, ..., S$, where S denotes the number of students. Due to the softmax activation, the output of the NN is a vector $\mathbf{p_{NN,M,s,t}}$ of dimension C (with C denoting the number of classes) for each student s, which represents the probability of each class.

4. EXPERIMENTAL EVALUATION

We evaluated the predictive performance of our classification pipeline on the collected data set (see Section 2). We conducted experiments to compare the performance of different model and feature combinations for the students' full data sequences as well as for early classification using partial sequences of the data, to answer our research questions.

Experimental Setup. We applied a train-test setting for all the experiments, i.e. parameters were fitted on a training data set and performance of the methods was evaluated on a test data set. Predictive performance was evaluated using the macro-averaged area under the ROC curve (AUC). We used the AUC as a performance measure as it is robust to class imbalance.

We performed all our experiments using different levels of detail for the classification task. As ground truth, we used the class labels presented in Fig. 2. Given the hierarchical nature of the decision tree separating the students in classes based on their conceptual knowledge, we performed the classification task focusing on three different levels of detail:

- *2-class case*: starting at the root of the decision tree (see Fig. 2), we divide students into two classes based on their understanding of the circuits: *both* (98 students) and *closed* (94 students).
- *3-class case*: going one step down in the hierarchy of the tree, we further divide the left branch of the tree (see Fig. 2) based on whether the students have completely understood all concepts (leading to a correct answer in our ranking task) or not. We therefore obtain three different classes: *correct* (38 students), *areasep* (60 students), and *closed* (94 students).
- *4-class case*: here, we also split the right branch of the tree (see Fig. 2) and divide the students into two groups based on whether they ranked the configurations in the task based on capacitance, resulting in four classes: *correct* (38 students), *areasep* (60 students), *capacitance* (47 students), and *other* (47 students).

For each of those three cases, we trained two types of classifiers (RF and NN) on our three different feature types (Action Counts - AC, Action Span - AS, Pairwise Embeddings - PW), using a stratified 10-fold **nested** cross validation. We kept the folds invariant across all experiments and stratified over the classes (according to the class labels of the 4-class case). Because of class imbalance, we used random oversampling for the training sets. We used a **nested** cross validation to avoid potential bias introduced by estimating model performance during hyperparameter tuning. This allowed us to tune the hyperparameters within the training folds (by further splitting them) and hold out the test sets for performance evaluation alone.

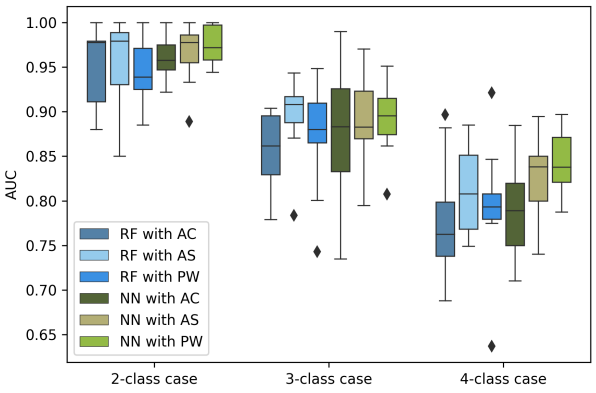


Figure 7: AUC for 4-class, 3-class and 2-class cases using different model and feature combinations. Predictions are made at the end of the interaction with the simulation, i.e. based on the complete sequential interaction data of the students.

For RF models, we tuned the following hyperparameters using a grid search: number of trees [5, 7, 9], number of features used at each decision level ['auto', 'all'], number of samples [bootstrap resampling of *training size* samples and balanced subsamples]. NN models were implemented using the *scikit-learn* library, trained for 300 epochs, and optimised for the log-loss function with the following hyperparameters: learning rate ['adaptive', 'invscaling'], initial learning rate [0.01, 0.001], solver ['adam', 'sgd'], hidden layer sizes and number [(32, 16), (64, 32), (64, 32, 16), (128, 64, 32, 16)], and activation function ['relu', 'tanh', 'identity'].

The skip-gram model providing our pairwise embedding features was implemented using the *TensorFlow* package. We trained the model for 150 epochs, with a window size of $w = 2$, a batch size of 16, and an embedding dimension of 15. We used categorical cross-entropy as the training loss. Because of its unsupervised nature, we trained the model on our whole dataset.

Offline Classification. In a first experiment, we were interested in assessing whether it is possible to associate students' behaviour in the simulation with their conceptual understanding achieved through the learning activity. This will be referred to as an offline classification task, since we are using students' complete interaction sequences r_s. The predictive performance in terms of AUC for the three classification problems (4-class case, 3-class case, and 2-class case) with a distinction between different model and feature combinations is illustrated in Fig. 7.

The results of this first experiment showed that for the 2-class case, all combinations of models and features reached very high average performances as quantified by their AUC scores (value range: $0.95 - 0.97$). The best mean score was achieved by the combination of NN with PW features ($AUC_{NN,PW} = 0.97$). However, it should be noted that the performance differences between the combinations were comparatively small. Using a one-way ANOVA, no statistically significant differences were found between the different groups ($F(5, 54) = 0.839, p = 0.528$). It seems that for this rather rough classification into groups of students who

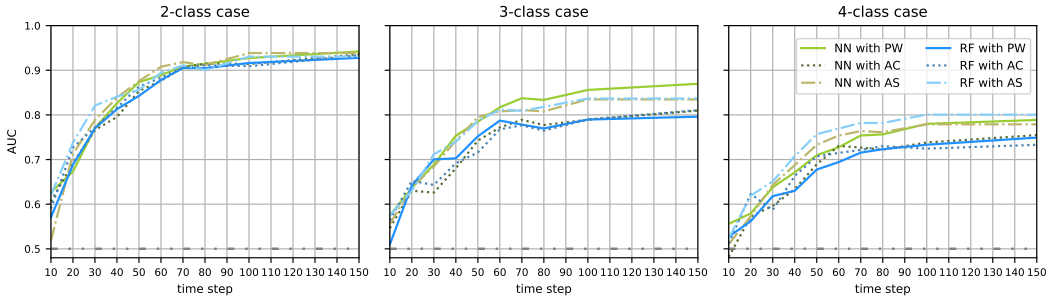


Figure 8: AUC for 4 classes, 3 classes, and 2 classes using different model and feature combinations. Predictions are made over time, stopping at time step $t = 150$ as only very few students have longer interaction sequences.

only understood the closed circuit and those who understood both the open and closed circuits, the different combinations of models and features perform equally well and with a high predictive accuracy.

By extending the classification task to the 3-class case, the AUC scores dropped in comparison to the 2-class case (value range: $0.86 - 0.90$). The lowest score was observed for RF with AC ($AUC_{RF,AC} = 0.86$), while best performances were observed for RF with AS ($AUC_{RF,AS} = 0.90$), NN with AS ($AUC_{NN,AS} = 0.89$), and NN with PW ($AUC_{NN,PW} = 0.89$). Similar to the 2-class case, no statistically significant differences were found between the different groups using a one-way ANOVA ($F(5, 54) = 0.740, p = 0.597$). This result illustrates that further dividing those who understood the functioning of the capacitor in both the open and closed circuit had a similar impact on the predictive performance of all model and feature combinations.

Finally, when evaluating the 4-class case, the most complex classification task, the mean AUC scores further dropped for all combinations (value range: $0.78 - 0.84$). The lowest performance was again observed for RF with AC ($AUC_{RF,AC} = 0.78$). Introducing the fourth class to the classification problem seemed to have a smaller negative impact on the AUC scores for NN with AS ($AUC_{NN,AS} = 0.83$) and NN with PW ($AUC_{NN,PW} = 0.84$), which obtain the best performances for the 4-class case. This observation was partially confirmed by a one-way ANOVA that showed a trend to statistical significance for differences between the combinations ($F(5, 54) = 1.989, p = 0.095$): as we increase the amount of classes, the p-value decreases.

The results of this first experiment show that it is possible to perform an offline prediction of students' conceptual understanding in the capacitor ranking task (see Section 2) based on the different combinations of models (NN or RF) and feature generation methods (AC, AS or PW) proposed in this work. From the entire data sequences of students' interactions with the simulation, we observed that predictive performance generally decreased when the complexity of the classification task was increased. While all combinations showed very good performances for the coarse classification of the 2-class case, AUC scores started to diverge more among combinations for the 3- and 4-class cases. Especially for the 4-class case, where differences became more visible with certain combinations showing a trend of better predictive performance (i.e. NN with PW and NN with AS) as compared to others (e.g., RF with AC).

Predicting over Time. In a second experiment, we were interested in assessing, whether we could predict students' conceptual knowledge for shorter interaction sequences, i.e. when not using students' full sequences, but only the first t interactions. For all three classification cases (2-class, 3-class, and 4-class), we trained all model (RF, NN) and feature (AC, AS, PW) combinations for $t = 10, 20, ..., 150$ time steps. As described in Section 3, to compute the features $\mathbf{f_{AC,s,t}}$, $\mathbf{f_{AS,s,t}}$, and $\mathbf{f_{PW,s,t}}$ for a student s at time step t, we only use the student's interactions i up to that point in time, i.e. $i = 1, ..., t$. Similarly, at each time step t, the models are exclusively used to predict students whose interactions sequences contain a minimum of t elements (i.e. $N_s \geq t$). For students with shorter interaction sequences (i.e. $N_s < t$), the last available prediction will be used. For example, for a student with 30 interactions ($N_s = 30$), we would make the first three predictions using the models for $t = 10$, $t = 20$, and $t = 30$ time steps. For the remaining time steps, the predictions from $t = 30$ will be carried over. We chose this approach for predicting because we assume that the student will leave the simulation after N_s interactions and it therefore does not make sense to update the prediction afterwards. Figure 8 illustrates the predictive performance in terms of AUC for the different model and feature combinations and all classification cases.

As expected, for the 2-class case, all models achieve a high performance for long interaction sequences. The AUC of all NN models is larger than 0.9, starting at time step $t = 70$. Generally, the difference between the models and feature combinations for $t \geq 70$ is small. We also observe some model differences for earlier time steps, where the RF model with the action span features performs better than the other models. It achieves an AUC larger than 0.8 already at time step $t = 30$ ($AUC_{RF,AS} = 0.82$). Moreover, the NN model with action span is close to that performance at time step $t = 30$ ($AUC_{NN,AS} = 0.79$). Naturally, predicting at even earlier time steps is more difficult, but some of the models achieve a decent AUC of 0.7 already after observing 20 interactions with the simulation ($AUC_{RF,AS} = 0.74$, $AUC_{RF,AC} = 0.73$, $AUC_{NN,AS} = 0.71$).

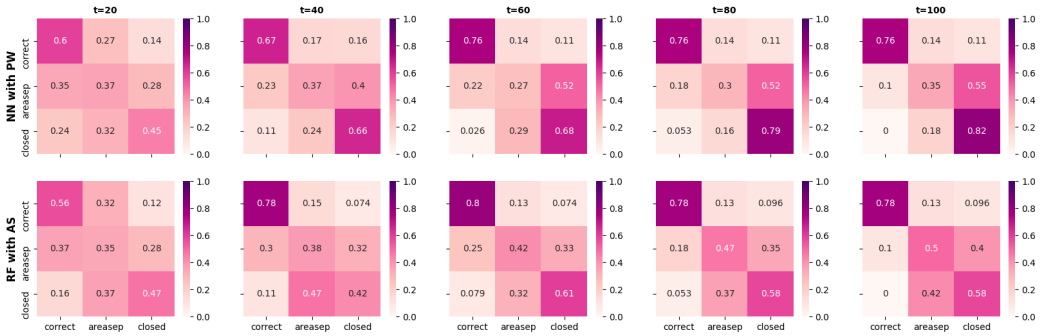


Figure 9: Evolution of confusion matrices over time steps for the 3-class case: NN with PW (top) and RF with AS (bottom).

For the 3-class case, the performances of all algorithms have decreased from time step $t = 10$ compared to the above problem. This is not surprising, as differentiating the correct students from the *areasep* students is not as straightforward as separating students who did not interact in the open circuit from the rest. Though, in the 2-class cases, all performances were close to one another, we notice that for the 3-class problem at time step $t = 40$, three model-features combinations take the lead ($AUC_{NN,PW} = 0.75$, $AUC_{RF,AS} = 0.74$, $AUC_{NN,AS} = 0.74$), while three fall behind ($AUC_{RF,PW} = 0.7$, $AUC_{RF,AC} = 0.69$ and $AUC_{NN,AC} = 0.68$) until $t = 70$, where the NN with PW outperforms all other model and feature combinations with an AUC of 0.87.

The variance across model-feature performances is larger in the 4-class case. From time step $t = 20$ already, RF with AS outperforms all other combinations, reaching an AUC of 0.8 at time step $t = 100$. Similarly to the 3-class problem, the same three models take the lead, while the others fall behind from time step $t = 70$ ($AUC_{NN,PW} = 0.75$, $AUC_{RF,AS} = 0.78$, $AUC_{NN,AS} = 0.76$ and $AUC_{RF,PW} = 0.72$, $AUC_{RF,AC} = 0.72$ and $AUC_{NN,AC} = 0.73$).

Given the fact the predictive performance in terms of AUC seems to be similar for the best performing models, we performed a more detailed analysis to assess how different the predictions of the several model and feature combinations were. In a real-world application (intervention setting), we would probably use a 2-class classifier to identify a coarse split (between classes *both* and *closed*) already at a relatively low number of time steps and use a 3-class method to provide a more detailed prediction later on. With the current model performance for the 4-class case, usage of a more detailed classifier seems not practicable. We therefore investigate the predictions of the two best models for the 3-class case: NN with PW and RF with AS. Figure 9 shows the confusion matrices for these two models for an increasing number of time steps. We do not show results for $t > 100$, as the predictive performance of the models does not improve much anymore for longer sequences (see also Fig. 8). While both models have a very similar predictive performance in terms of AUC up to time step 60, we can already see that the models evolve differently in terms of prediction. At time step $t = 40$, the RF model is already very accurate in detecting students from class *correct* ($\hat{p}(correct|c_{true} = correct) = 0.78$), while the NN model is less confident ($\hat{p}(correct|c_{true} = correct) = 0.67$). On the other hand, the NN model is already more accurate in identifying students from class *closed* ($\hat{p}(closed|c_{true} = closed) = 0.66$), while the RF model cannot identify students from this class well ($\hat{p}(closed|c_{true} = closed) = 0.42$). At time step $t = 60$, both models are almost equally accurate in identifiying students from class *correct* (NN: $\hat{p}(correct|c_{true} = correct) = 0.76$, RF: $\hat{p}(correct|c_{true} = correct) = 0.80$). The NN model is still better at classifying students from the class *closed*. Both models have trouble with correctly identifying students from class *areasep*. While the NN model tends to assign these students to class *closed* ($\hat{p}(areasep|c_{true} = closed) = 0.52$), the RF model is becoming better at correctly assigning them ($\hat{p}(closed|c_{true} = closed) = 0.42$). These observed trends continue to get stronger with an increasing number of time steps. At $t = 100$, the NN model is very accurate when it comes to classifying students from classes *correct* and *closed*. Students from class *areasep* have only a 35% chance of being correctly classified and a 55% chance of getting assigned to class *closed*. In practice, this would mean that 55% of the students would get more intervention (hints) than necessary. The RF classifier is also very accurate in detecting students from class *correct*, but is, however, not able to distinguish between students from class *closed* and class *areasep*. In practice, this would mean misclassified students from class *closed* would get less help than necessary and misclassified students from class *areasep* would get more help than necessary.

This experiment shows that we can (coarsely) classify students after observing a relatively low number of interactions. For the 2-class case, the AUC of the best model (RF with AS) is larger than 0.8 after $t = 30$ time steps. Naturally, the classification task is more complex for the 3-class and the 4-class cases. The best model on the 3-class case (NN with PW) achieves an AUC close to 0.8 at time step 50. The second analysis demonstrates that achieving a similar predictive performance in terms of AUC does not imply the same classification behaviour, i.e.the best models on the 3-class case (NN with PW and RF with AS) have different strengths. It has, however, one important limitation: students spend different amount of times on the simulation and therefore, the length of their interaction sequences varies. There are for example students with 80 interactions and other students with only 50 interactions. Performing the classification task at time step 40 is early for a student with

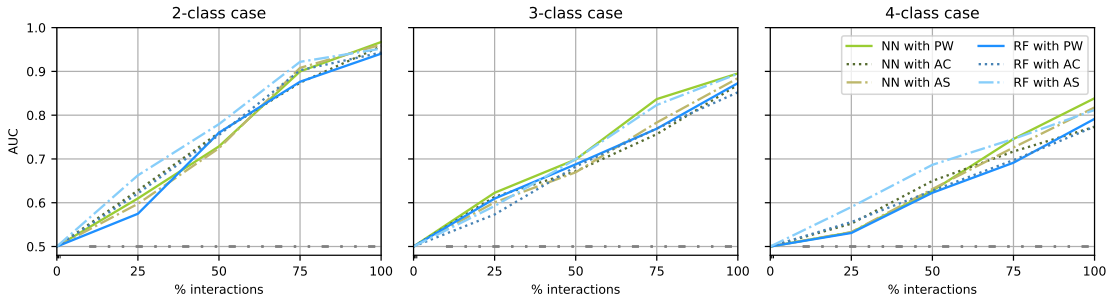


Figure 10: AUC on the 2-class, 3-class and 4-class cases for different model and feature combinations. Predictions were made for 25%, 50%, 75% and 100% of the total number of interactions for each student.

a total of 80 interactions. For a student with only 50 interactions in total, the prediction at time step 40 comes almost at the end of the interaction time with the simulation.

Online (Early) Classification. The third and last experiment addresses the limitations of the previous experiment. In this experiment, we were interested in assessing the "early" predictive performances of the different models using only a part of a student's sequence. Given the fact that the length of interaction sequences varies over the students, we did not align the sequences by absolute time steps, but by percentages of interactions. Specifically, we aimed at making the prediction for a student after having seen 25%, 50%, 75%, and 100% of the interaction sequence of this student. Note that this experiment does not require a re-training of our models. We just retrieve the predictions of the models for the corresponding time step t. For our example student with a total of 80 interactions, we retrieve the predictions of the models for time steps $t = 20$, $t = 40$, $t = 60$, and $t = 80$. Figure 10 shows the AUC of the models for all classification cases, with an increasing number of interactions (in %).

As expected, all model and feature combinations perform well for the 2-class case. To achieve a high classification accuracy, we do not need to observe the full sequence of a student. For all NN models, obtaining 75% of students' interactions is enough to achieve an AUC of around 0.9. With RF, the model using the action span features also obtains an AUC of more than 0.9 at 75% of the interactions ($AUC_{RF,AS} = 0.92$). The performance of the two other feature types is slightly lower ($AUC_{RF,AC} = 0.9$, $AUC_{RF,PW} = 0.88$). Naturally, predictive performance of all the models is lower when observing smaller parts of students' interaction sequences. If obtaining only the first 25% of students' interactions, there is more variation in the achieved AUC between models, with the best model (RF with AS) achieving an AUC of 0.66 and the worst model achieving an AUC of 0.57 (RF with PW). It is promising that the best model at 50% of the interactions exhibits an AUC of almost 0.8 ($AUC_{RF,AS} = 0.78$), which makes it a valuable candidate for a coarse early prediction and intervention, i.e. differentiating between students with a high conceptual understanding (class *both*) and students with a low conceptual understanding (class *closed*), early on.

Naturally, performance of the models for the 3-class case is overall lower as we are now differentiating the different levels of conceptual knowledge in a more fine-grained way. As we have seen in Fig. 9), it is difficult to differentiate between students from the left branch of the tree (i.e. *correct* vs. *areasep* in Fig. 2). Performance across models varies more for the 3-class case. When observing 75% of students interactions, the AUC of the worst model (NN with AS) amounts to 0.78, while the best model (NN with PW) has an AUC of 0.84. We also observe that the NN with PW features is consistently the best model, regardless of the amount of observed interactions, with an increasing gap to the other models. At 50% of interactions, the gap in performance among the three best models is still small ($AUC_{NN,PW} = 0.699$, $AUC_{RF,AS} = 0.7$, $AUC_{RF,PW} = 0.69$). It gets larger at 75% ($AUC_{NN,PW} = 0.84$, $AUC_{RF,AS} = 0.82$, $AUC_{NN,AS} = 0.78$). When observing the complete sequences of the students (i.e. 100% of the interactions), all models reach an AUC of 0.85 or higher (see also Fig.7).

Again, the performance decreases when moving to four classes, due to the increasing complexity (in terms of level of detail) of the classification task. The AUC of the best model (NN with PW) amounts to 0.83, while for the worst two models $AUC = 0.77$ (RF with AC, NN with AC). While all the models' AUC is lower than 0.7 when observing only 25% or 50% of students' interactions, interestingly there is a large gap between the best model (RF with AS) and all the other models (i.e. at 25%: $AUC_{RF,AS} = 0.59$, $AUC_{RF,AC} = 0.56$).

With this last experiment, we assessed the capabilities of our models to make predictions as early as possible during interaction with the simulation as a basis for intervention. By evaluating the models at different percentages of total interactions, we took into account the fact that the definition of 'early' depends on the student. Our results show that after observing the first 50% of interactions, we are able to reliably distinguish between students with a high and low conceptual understanding gained by the end of the learning activity (*both* and *closed*). At 75% of interactions, the best models are also able to provide a more fine-grained prediction (*correct*, *areasep*, and *closed*).

5. DISCUSSION

Over the last decade, interactive simulations of scientific phenomena have become increasingly popular. They allow students to learn the principles underlying a domain through

their own explorations. However, only few students possess the degree of inquiry skills and self-regulation necessary for effective learning in these environments. In this paper, we therefore explored approaches for an early identification of struggling students as a basis for adaptive guidance: we aimed at predicting students' conceptual knowledge while interacting with a Physics simulation. Specifically, we were interested in answering the following three research questions: 1) Can students' interaction with the data be associated with the gained conceptual understanding? 2) Can conceptual understanding be inferred through sequence mining methods with embeddings? 3) Can the proposed methods be used for early predicting students' conceptual understanding based on partial sequences of interaction data?

To answer the first research question, we analysed data from 192 first-year undergraduate Physics students who used an interactive capacitor simulation to solve a task in which they had to rank four capacitor configurations by their stored energy. Previous research has emphasised the importance of aligning instructional and assessment activities to implement pedagogically meaningful learning activities with educational technology [26, 27]. Since one objective of automatically detecting student learning behavior is to provide some kind of assessment (either formative or summative), the learning task presented in this work was designed to facilitate the application of sequence mining. The design of this learning task allowed us to relate all of the students answers to a certain level of conceptual understanding. Using a decision tree, we were able to map each ranking to one of the four labels representing groups of similar conceptual understanding. Our results show that all evaluated models were able to correctly associate students' sequential interaction data in the simulation with the generated labels, achieving high predictive performance when fed with full sequences (2-class case: $AUC > 0.9$, 3-class case: $AUC > 0.85$, 4-class case: $AUC > 0.75$). This high predictive power was also observed in [22], where they reached an accuracy of 85% when separating 'high' learners from 'low' learners based of full interaction sequences. However, despite their findings of a potential third cluster, they did not investigate the ternary classification task. In this paper, we increase the granularity of the labels in order to target more specific shortcomings in the knowledge of the students in order to provide them with more detailed feedback. We therefore conclude that we can answer research question 1) with yes.

The second research question investigates the benefits of latent features generated by skip-grams for offline classification tasks in the context of education. Usually applied to NLP problems, skip-grams have the ability to learn the context of a word in an unsupervised fashion. In our case, we use it to find the OELE behaviour of the students surrounding their interaction, and retrieve the embedding matrix of the neural network to create our latent representations. This approach has already been proven efficient to analyse student strategies in blended courses [28], but not for the identification of conceptual understanding. To evaluate the predictive power of latent feature representations, we trained two classifiers (NN and RF) on three types of feature (*Action Counts*, *Action Span* and *Pairwise Embeddings*) on the full sequences of students. At first, we notice that all model and feature combinations achieve a high AUC for all classification tasks (2-class case, 3-class case, and 4-class case). Though the ANOVA revealed no significant differences between the predictive performance of models with different types of features, we can observe that the NN with PW achieves a higher performance on average than all other combinations but the NN with AS. What is more, the performances in its first quartile dominate those from the third quartile of three model and feature combinations. Additionally, its performance variance is smaller than that of the NN with AS. This shows that pairwise embeddings generated by a skip-gram approach can be a valuable asset for finer-grained classification, even if no statistical difference was found with respect to the other model-feature combinations. We can therefore answer research question 2) with a partial yes.

To address the third research question, we assessed predictive performance of the proposed approach when only partial sequences of the students' interaction data were observed. We analysed the performances of our proposed approaches based on varying proportions of the available data and for classification tasks with different levels of complexity. The results of our experiments show that the proposed combinations of models and generated features allowed us to predict the correct class labels early on. The best models were able to reliably predict students' conceptual understanding for the 2-class case ($AUC \approx 0.8$) after having seen 50% of the students' interaction data. To reach a similar predictive performance for the more fine-grained 3-class and 4-class cases, the best models needed about 75% of the data. The findings from these experiments therefore represent a promising step towards early prediction of students' conceptual understanding in OELEs. We can therefore answer research question 3) with yes.

One of the limitations of this work is the unfeasibility to track whether students used external resources (other than the simulation) in order to rank the four capacitor configurations. This may bias the inference from the simulation usage to the extrapolated understanding level. Furthermore, due to our small sample size, we were able to only train shallow NN classifiers and skip-grams. Finally, the external validity of these experiments remains to be evaluated on other interactive simulations and different types of tasks.

To conclude, the proposed approach represents a promising step towards early prediction of students' learning strategies in interactive simulations, that moreover, can be associated with their level of conceptual understanding. The proposed learning activity seems to represent an interesting example for the design of learning tasks in OELEs that facilitates the association of detected student strategies with conceptual understanding through sequence mining. Future work could explore whether such designs could also be used to identify conceptual understanding at a more fine-grained level.

6. ACKNOWLEDGMENTS

We thank Kathy Perkins (PhET Interactive Simulations, University of Colorado Boulder) and Daniel Schwartz (Stanford University) for the valuable input on the study design and setup and for supporting data collection.

References

[1] C. E. Wieman, W. K. Adams, and K. K. Perkins, "PhET: Simulations That Enhance Learning," *Science*, vol. 322, no. 5902, pp. 682–683, 2008.

[2] X. Fan and D. Geelan, "Enhancing students' scientific literacy in science education using interactive simulations: A critical literature review," *Journal of Computers in Mathematics and Science Teaching*, vol. 32, no. 2, pp. 125–171, 2013.

[3] R. E. Mayer, "Should there be a three-strikes rule against pure discovery learning? the case for guided methods of instruction," *American Psychologist*, vol. 59, no. 1, pp. 14–19, 2004.

[4] L. Alfieri, P. J. Brooks, N. J. Aldrich, and H. R. Tenenbaum, "Does discovery-based instruction enhance learning?" *Journal of Educational Psychology*, vol. 103, no. 1, pp. 1–18, 2011.

[5] P. A. Kirschner, S. J., and C. R.E., "Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching," *Educational Psychologist*, vol. 41, no. 2, pp. 75–86, 2006.

[6] I. Roll, V. Aleven, and K. R. Koedinger, "The invention lab: Using a hybrid of model tracing and constraint-based modeling to offer intelligent support in inquiry environments," in *Proc. ITS*, 2010, pp. 115–124.

[7] S. Perez, J. Massey-Allard, D. Butler, J. Ives, D. Bonn, N. Yee, and I. Roll, "Identifying productive inquiry in virtual labs using sequence mining," in *Proc. AIED*, 2017, pp. 287–298.

[8] C. Brand, J. Massey-Allard, S. Perez, N. Rummel, and I. Roll, "What inquiry with virtual labs can learn from productive failure: A theory-driven study of students' reflections," 2019, pp. 30–35.

[9] R. Baker, J. Clarke-Midura, and J. Ocumpaugh, "Towards general models of effective science inquiry in virtual performance assessments," *Journal of Computer Assisted Learning*, vol. 32, no. 3, pp. 267–280, 2016.

[10] E. Bumbacher, S. Salehi, M. Wierzchula, and P. Blikstein, "Learning Environments and Inquiry Behaviors in Science Inquiry Learning: How their Interplay Affects the Development of Conceptual Understanding in Physics," *Proc. EDM*, pp. 61–68, 2015.

[11] S. Perez, J. Massey-Allard, J. Ives, D. Butler, D. Bonn, J. Bale, and I. Roll, "Control of variables strategy across phases of inquiry in virtual labs," pp. 271–275, 2018.

[12] J. D. Gobert, M. S. Pedro, J. Raziuddin, and R. S. Baker, "From Log Files to Assessment Metrics: Measuring Students' Science Inquiry Skills Using Educational Data Mining," *Journal of the Learning Sciences*, vol. 22, no. 4, pp. 521–563, 2013.

[13] D. Tavares, K. Perkins, M. Kauzmann, and C. Velez, "Towards a teacher dashboard design for interactive simulations," in *Journal of Physics: Conference Series*, vol. 1287, no. 1, 2019, p. 012055.

[14] R. Sawyer, J. Rowe, R. Azevedo, and J. Lester, "Filtered time series analyses of student problem-solving behaviors in game-based learning." *Proc. EDM*, 2018.

[15] J. Sabourin, J. Rowe, B. Mott, and J. Lester, "Exploring affect and inquiry in open-ended game-based learning environments," in *Proc. ITS (Workshop on Emotions in Games for Learning)*, 2012, pp. 1–8.

[16] E. Rowe, R. Baker, J. Asbell-Clarke, E. Kasman, and W. Hawkins, "Building Automated Detectors of Gameplay Strategies to Measure Implicit Science Learning," in *Proc. EDM*, 2014, pp. 337–338.

[17] M. Eagle, E. Rowe, D. Hicks, R. Brown, T. Barnes, J. Asbell-Clarke, and T. Edwards, "Measuring Implicit Science Learning with Networks of Player-Game Interactions," in *Proc. CHI in Play*, 2015, pp. 499–504.

[18] T. Käser and D. Schwartz, "Modeling and analyzing inquiry strategies in open-ended learning environments," *IJAIED*, vol. 30, no. 3, pp. 504–535, 2020.

[19] S. Kardan and C. Conati, "A framework for capturing distinguishing user interaction behaviours in novel interfaces," *Proc. EDM,*, pp. 159–168, 2011.

[20] T. Käser, A. G. Busetto, B. Solenthaler, J. Kohn, M. von Aster, and M. Gross, "Cluster-Based Prediction of Mathematical Learning Patterns," in *Proc. AIED*, 2013, pp. 389–399.

[21] S. Amershi and C. Conati, "Combining Unsupervised and Supervised Classification to Build User Models for Exploratory Learning Environments," 2009, pp. 18–71.

[22] L. Fratamico, C. Conati, S. Kardan, and I. Roll, "Applying a framework for student modeling in exploratory learning environments: Comparing data representation granularity to handle environment complexity," *IJAEID*, vol. 27, no. 2, pp. 320–352, 2017.

[23] T. Käser and D. L. Schwartz, "Exploring Neural Network Models for the Classification of Students in Highly Interactive Environments," *Proc. EDM*, pp. 109–118, 2019.

[24] K. V. Ghag and K. Shah, "Comparative analysis of effect of stopwords removal on sentiment classification," in *Proc. IC4*. IEEE, 2015, pp. 1–6.

[25] Y. Goldberg and O. Levy, "Word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *CoRR*, vol. abs/1402.3722, 2014.

[26] T. Lauwers, "Aligning capabilities of interactive educational tools to learner goals," Ph.D. dissertation, CMU, Pittsburgh, PA, May 2010.

[27] C. Giang, "Towards the alignment of educational robotics learning systems with classroom activities," Ph.D. dissertation, EPFL, Lausanne, CH, 2020.

[28] N. Akpinar, A. Ramdas, and U. Acar, "Analyzing student strategies in blended courses using clickstream data," in *Proc. EDM*, 2020, pp. 6–17.

Knowing *When* and *Where*: Temporal-ASTNN for Student Learning Progression in Novice Programming Tasks

Ye Mao
North Carolina State Univ.
Raleigh, NC, USA
ymao4@ncsu.edu

Yang Shi
North Carolina State Univ.
Raleigh, NC, USA
yshi26@ncsu.edu

Samiha Marwan
North Carolina State Univ.
Raleigh, NC, USA
amarwan@ncsu.edu

Thomas W. Price
North Carolina State Univ.
Raleigh, NC, USA
twprice@ncsu.edu

Tiffany Barnes
North Carolina State Univ.
Raleigh, NC, USA
tmbarnes@ncsu.edu

Min Chi
North Carolina State Univ.
Raleigh, NC, USA
mchi@ncsu.edu

ABSTRACT

As students learn how to program, both their programming code and their understanding of it evolves over time. In this work, we present a general data-driven approach, named *Temporal-ASTNN* for modeling student learning progression in open-ended programming domains. Temporal-ASTNN combines a novel neural network model based on abstract syntactic trees (AST), named ASTNN, and Long-Short Term Memory (LSTM) model. ASTNN handles the *linguistic* nature of student programming code, while LSTM handles the *temporal* nature of student learning progression. The effectiveness of ASTNN is first compared against other models including a state-of-the-art algorithm, Code2Vec across two programming domains: iSnap and Java on the task of program classification (*correct* or *incorrect*). Then the proposed temporal-ASTNN is compared against the original ASTNN and other temporal models on a challenging task of student success early prediction. Our results show that Temporal-ASTNN can achieve the best performance with only the first 4-minute temporal data and it continues to outperform all other models with longer trajectories.

Keywords

Student Modeling in Programming, LSTM, ASTNN

1. INTRODUCTION

Learning how to program is like learning how to write in a second language. As students learn to author code, both their programming code and their understanding of it evolves over time. Prior research has either focused *exclusively* on developing accurate *linguistic* models of their artifacts [30, 24, 1, 42], or developing *temporal* models of students comprehension of programming [11, 21, 23]. In this work, we propose a general data-driven approach named *Temporal-ASTNN*, which combines a state-of-the-art neural network model based on abstract syntax trees (AST) named ASTNN – addressing the *linguistic structure* of the students' artifacts – along with Long-Short Term Memory (LSTM), which handles their *learning progression*. In this way we effectively marry *both* aspects of the process in a single system.

Much as *language* is how people communicate, *programming languages* are how we communicate with machines, and various natural language processing (NLP) techniques can be applied to modeling programming languages [15]. Traditional approaches for code representation often treat code fragments as natural language texts and model them based on their tokens [7, 9]. Despite their simplicity, token-based methods omit the rich and explicit *structural* information [25] in student codes. Until recently, deep learning models have achieved state-of-the-art results on source code analysis, including code functionality classification [24], method name prediction [1], code clone detection [42] and so on. These successful models usually combine Abstract Syntax Tree (AST) representations with various neural networks to capture the structural information from the programming language. Their impressive performance shows that by addressing the *linguistic structural* nature of code, syntactic knowledge is indeed important to learn meaningful code representation.

On the other hand, modeling student learning progression in open-ended programming environments is also a type of student modeling. Generally speaking, student modeling has been widely applied to predict the student's future performance based on historical data. For well-defined learning environments, student models usually monitor students' learning progress (*correct* or *incorrect*) over time to infer their knowledge states, such as Bayesian Knowledge Tracing (BKT) [8] and Deep Knowledge Tracing (DKT) [29]. When it comes to open-ended programming environments, student modeling becomes much more challenging because 1) the correctness evaluation concerning each step taken by students will not be available, and 2) it is extremely hard to represent student states. As a result, prior research either has focused on utilizing other features such as hint usage, interface interactions to evaluate student learning outcomes [11], or creating meaningful states by transforming student click-like log files into fixed feature sets for various student

Ye Mao, Yang Shi, Samiha Marwan, Thomas Price, Tiffany Barnes and Min Chi "Knowing When and Where: Temporal-ASTNN for Student Learning Progression in Novice Programming Tasks". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 172-182. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

modeling tasks [21]. While such prior work is able to capture the *temporal* information from historical data, it ignores the *linguistic, structural* property of student code. As an accurate student model is a building block for any educational system that provides adaptivity and personalization, it is especially important to model student learning progression in open-ended programming tasks by addressing both *linguistic* and *temporal* characteristics in student code sequences.

In this work, we present a data-driven approach named *Temporal-ASTNN* to model student learning progression in open-ended programming domains. Temporal-ASTNN consists of two main modules: 1) ASTNN [42] for code representation learning, which can handle the *linguistic* structure of student code, and 2) LSTM [16] for temporal learning, which handles the *temporal* nature of student learning progression. In order to explore the effectiveness of our model, we focus on two types of student modeling tasks. One is *the task of program classification (correct or incorrect)*, in which the effectiveness of ASTNN is compared against other models including a state-of-the-art algorithm, Code2Vec [1] across two programming domains: an open-ended block-based programming environment named iSnap and a textual programming environment for the Java programming language. The other is the task of *student success early prediction* in which the effectiveness of temporal-ASTNN is compared against the original ASTNN and other models integrating with different feature embeddings on iSnap only because it has trajectories of student codes.

Our main contributions are: 1) To the best of our knowledge, Temporal-ASTNN is the first model to address both *linguistic* and *temporal* properties of student learning progression in programming tasks; 2) We explored the robustness and the effectiveness of our model on student success early prediction task and compared it with state-of-the-art temporal models; and 3) We evaluated the effectiveness of ASTNN against Code2Vec and various baseline models on student program classification tasks across two domains, while most prior research mainly focused on classic tasks of *professional* source code analysis instead of novice programming.

The remainder of this paper is structured as follows. Section 2 presents the methods. Section 3 and 4 describe the two types of programming tasks together with experimental settings and results. Section 5 presents the related work. Finally, we discuss and conclude our work in Section 6.

2. METHODS

Problem Definition: For the task of student program classification, our dataset can be represented $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \{\langle \boldsymbol{x}^1, y^1 \rangle, \langle \boldsymbol{x}^2, y^2 \rangle, ..., \langle \boldsymbol{x}^N, y^N \rangle\}$ where N is the total number of codes in the dataset where $\boldsymbol{x}^i$ represents a code snippet of student i and binary y^i indicates whether the code is correct or not.

For the task of student success early prediction, our dataset can be represented as $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^M\}$, where M is the number of students. For a given student k, $\boldsymbol{x}^k = \{\boldsymbol{x}^k_1, ..., \boldsymbol{x}^k_{T_k}\}$, where $\boldsymbol{x}^k_t$ represents student k's code at time step t in $\boldsymbol{x}^k$ and T_k is the total number of codes in the student k's learning trajectories which varies with different students. For each $\boldsymbol{x}^k$, we are provided with the outcome label y^k for the outcome of the sequence of codes. $y^k = 0$ indicates the student

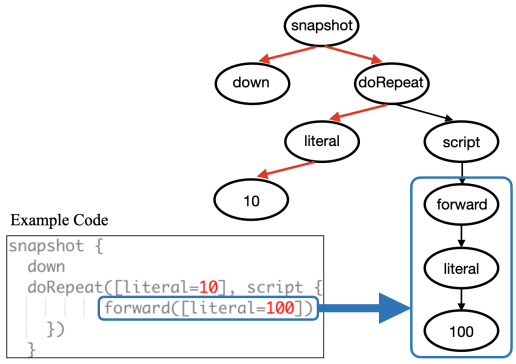


Figure 1: An examaple of iSnap code and the AST representing its syntactic structure. Red highlights a sample path, and blue highlights a sample ST-tree.

k succeeded, otherwise $y^k = 1$. The goal of student success early prediction is to predict the y^k using the student's codes from the beginning up to the certain minutes: $\boldsymbol{x}^k_1, \boldsymbol{x}^k_2, ..., \boldsymbol{x}^k_t$. For simplicity, we omit index k hereinafter when it does not cause ambiguity.

2.1 Temporal-ASTNN

Figure 2 shows the detailed structure of Temporal-ASTNN. Fundamentally, it contains a ASTNN which learns the embedding for student code and a LSTM layer which handles the temporal aspect. It is important to note that in Temporal-ASTNN, the two modules interact with each other to control how information flows.

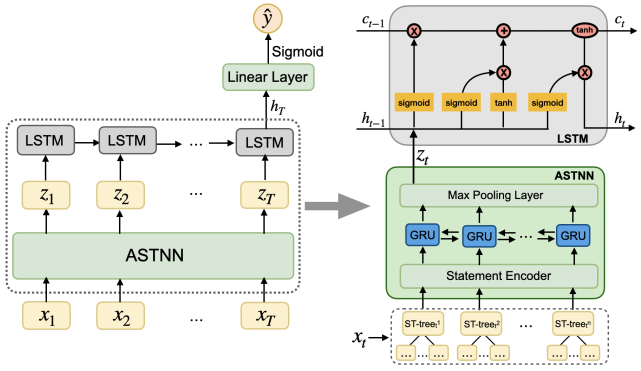


Figure 2: Temporal-ASTNN model structure: the output of ASTNN connects to the input of LSTM.

2.1.1 ASTNN

ASTNN is one of the state-of-the-art methods in source code analysis, and it's main idea is to learn a vector for the code through statement-level ASTs. Specifically, we split the large AST of a code fragment by the granularity of statement and extract a sequence of statement trees (ST-trees) via pre-order traversal. As shown in Figure 1 (highlighted in blue), we can get a ST-tree rooted at `forward`, whose child is `literal` and grandchild is `100`. In this way, we will get a sequence of ST-trees from the original AST, and feed them as the raw input of ASTNN. As shown in Figure 2, ST-trees

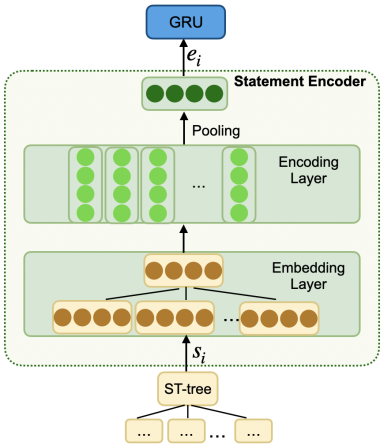


Figure 3: Statement Encoder of ASTNN, composing by an Embedding layer, an Encoding layer and a max-pooling layer.

will first pass Statement Encoder, then go through Bidirectional GRU (Bi-GRU) [2], finally pass max-pooling layer to get the vector for code representation.

Statement Encoder: Figure 3 shows the detailed structure of statement encoder. Assuming that there are J total nodes in a ST-tree s_i, for each node $n_i^j \in s_i$, $j \in [1, J]$, it will first go through the embedding layer to get initial embeddings $v_i^j = \mathbf{W_{embed}}^\top n_i^j$, where $\mathbf{W_{embed}} \in \mathbb{R}^{V \times d}$ is the pre-trained embedding matrix, V is the vocabulary size and d is the embedding dimension. Then the vector will be updated through a Recursive Neural Network [35] based encoder layer: $h_i^j = \sigma(\mathbf{W_{encode}}^\top v_i^j + \sum h_{\text{child}} + b_i^j)$. Here $\mathbf{W_{encode}} \in \mathbb{R}^{d \times k}$ is the encoding matrix and k is the encoding dimension. b is the biased term and σ is the activation function, in this work we followed the original paper to set σ as identify function. After recursive optimization of the vectors of all node in the ST-tree, we sample the final representation e_i via a max-pooing layer:

$$e_i = [max(h_i^{j_1}), max(h_i^{j_2}), ..., max(h_i^{j_k})], j \in [1, J] \quad (1)$$

Code Representation: For a set of ST-trees $(s_1, s_2, ..., s_L)$, where L is the number of ST-trees in the AST, our goal is to get a code vector z as final representation. After generating a sequence of vectors $(e_1, e_2, ..., e_L)$ from Statement Encoder, we will apply Bi-GRU to track the naturalness of statements sequence:

$$h_i = [\overrightarrow{GRU}(e_i), \overleftarrow{GRU}(e_i)], i \in [1, L] \quad (2)$$

The statement representation $h_i \in \mathbb{R}^{L \times 2m}$, where m is the embedding dim of Bi-GRU. Finally, similar to Statement Encoder, a max-pooling layer is used to sample the most important features on each of the embedding dimension. Thus we get $z \in \mathbb{R}^{2m}$, which is treated as the final vector representation of the original code fragment.

In original ASTNN, we can add another linear layer to directly fit z to the following prediction tasks. While in Temporal-ASTNN, z will be used as the input for LSTM memory cell.

2.1.2 LSTM

As shown in Figure 2, at each time step t, the output of ASTNN z_t will be used as the input for LSTM cell. Once ASTNN generates the code representation by learning the *linguistic* nature from code x_t:

$$z_t = \text{ASTNN}(x_t) \quad (3)$$

LSTM is trained utilizing input vector z_t to handle the temporal information. There are three major components: a forget gate, an input gate, and an output gate in a LSTM memory cell.

Forget Gate: In the first step, a function of the previous hidden state h_{t-1} and the new code input z_t passes through the forget gate, indicating what is probably irrelevant and can be taken out of the cell state. The forget component will calculate a weight f_t between 0 to 1 for each element in hidden state vector C_{t-1}. Here W_f and b_f are the weights and bias for the forget component.

$$f_t = \text{sigmoid}(\mathbf{W_f} \cdot [h_{t-1}, z_t] + b_f) \quad (4)$$

Input Gate: There are two steps involved in input component's calculation. In the first step, a *tanh* layer calculates a candidate vector $\tilde{C}_t$ that could be added to the current hidden state. In the second step, the input components calculate a weight vector i_t (ranging from 0 to 1) to determine to what extent $\tilde{C}_t$ should update the current memory state.

$$\begin{aligned} \tilde{C}_t &= \tanh(\mathbf{W_c} \cdot [h_{t-1}, z_t] + b_c) \\ i_t &= \text{sigmoid}(\mathbf{W_i} \cdot [h_{t-1}, z_t] + b_i) \end{aligned} \quad (5)$$

Output Gate: The output component is simply an activation function that filters elements in memory cell state C_t, where $C_t = C_{t-1} \cdot f_t + \tilde{C}_t \cdot i_t$. It calculates a weight vector to determine how much information is allowed to be revealed:

$$o_t = \text{sigmoid}(\mathbf{W_o} \cdot [h_{t-1}, z_t] + b_o) \quad (6)$$

Finally we get the output of time t: $h_t = o_t * \tanh(C_t)$. In this work, we used the last-step output from LSTM as the temporal representation of student code sequence.

2.1.3 Temporal-ASTNN: Truncated vs. Entire

As shown in Figure 2, by combining ASTNN and LSTM, the final Temporal-ASTNN can be described as:

$$\begin{aligned} z_1, ..., z_T &= \text{ASTNN}(x_1, ..., x_T) \\ h_T &= \text{LSTM}(z_1, ..., z_T) \\ \hat{y} &= \text{sigmoid}(\mathbf{W_l} h_T + b_l) \end{aligned} \quad (7)$$

where $\hat{y}$ is the output from Temporal-ASTNN, $\mathbf{W_l}$ is the weight matrix b_l is the bias term for the liner layer. The entire Temporal-ASTNN framework is learned by optimizing ASTNN and LSTM parameters spontaneously. They are optimized by minimizing the binary cross-entropy:

$$\mathcal{L}(\hat{y}, y; \Theta) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \quad (8)$$

Prior research on applying ASTNN for source code analysis only used one snippet of code fragment to extract meaningful representation for following machine learning tasks. However, when combining ASTNN with LSTM on student

programming sequences such as iSnap for early prediction, we have the choices of either using the truncated training sequences or using the entire sequences. The advantage of using truncated sequences is that the training data would be more similar to the testing data and thus, the learned representations are more likely to emerge and be representative for the early success task. On the other hand, the advantage of using *entire* sequences is that the longer the sequences, the more meaningful AST patterns can be considered and discovered. Thus, we explored Temporal-ASTNN using both the *entire* and the *truncated* sequences for representation learning and referred as Temporal-ASTNN$_{\text{Trunc}}$ and Temporal-ASTNN$_{\text{Entire}}$ respectively.

2.2 Code2Vec

Code2Vec [1] leverages different features and model structures, and focuses on the dependency of distant components in code structures to achieve code classification tasks. As with ASTNN, Code2Vec is designed to address the *linguistic* structure of programming languages. Fundamentally, there are two main differences between these two models: 1) ASTNN takes a set of statement-level ASTs as inputs, while Code2Vec utilizes the syntactic paths of ASTs to learn the representation (an example path is shown in Figure 1 in red). And 2) After encoding the vector representations of ST-trees, ASTNN uses Bi-GRU to handle the sequence of vectors; while Code2Vec utilizes an attention mechanism to learn a weighted average of path vectors and thus to produce the final code representation. With the vector representing code, Code2Vec can also be used for various prediction tasks.

3. STUDENT PROGRAM CLASSIFICATION

In the task of student program classification, we aimed to predict the correctness (correct or incorrect) of student submitted code. The effectiveness of ASTNN is compared against Code2Vec and other token-based models across two programming domains: iSnap and Java.

3.1 Datasets

3.1.1 iSnap

iSnap is an extension to Snap*!* [13], a block-based programming environment, used in an introductory computing course for non-majors in a public university in the United States [32]. iSnap extends Snap*!* by providing students with data-driven hints derived from historical correct student solutions [31]. In addition, iSnap logs all students actions while programming (e.g. adding or deleting a block), as a *trace*, allowing us to detect the sequences of all student steps, as well as the *time* taken for each step. In this work, we focused on one homework exercise named Squiral, derived from the BJC curriculum [13]. In Squiral, students are asked to write a procedure that draws a square-like spiral. As shown in Figure 4, correct solutions require procedures, loops, and variables using at least 7 lines of code. We collected students' data for Squiral from Spring 2016, Fall 2016, Spring 2017, and Fall 2017. We excluded students who requested hints from iSnap to eliminate factors that might affect students' problem-solving progress, leaving a total of 65, 38, 29, and 39 student code traces from each semester, respectively.

The data collected from iSnap consists of a code trace for each student's attempt. This code trace represents a sequence of timestamped snapshots of student code. In prior research, an expert feature detector has been proposed to automatically detect 7 expert features of a student snapshot [43]. Those expert features are binary and indicate whether the corresponding feature presents or not. We ran the expert-feature detector to tag each snapshot in all 171 code traces, making a total of 31,064 tagged snapshots. With the temporal sequences, iSnap data is evaluated not only on this classification task, but also on the temporal early prediction task as described in Section 4.

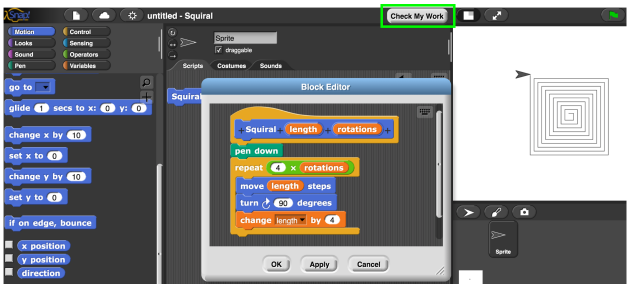


Figure 4: The iSnap interface, with the blocks palette on the left, the output stage on the right, the scripting area in the middle, and the hints button on top.

3.1.2 CodeWorkout

CodeWorkout[1] is an online and open system for programming in Java. It provides a web-based platform on which students from various backgrounds can practice programming and instructors can offer courses [10]. Different from iSnap, CodeWorkout *doesn't* log students' traces during programming but only their submissions. In this work, we focused on one programming exercise named isEverywhere, where the knowledge of loops and array will be mainly evaluated. In isEverywhere, students are asked to write a Java function to check if a value is "everywhere", that is in the given array if the value exists for every pair of adjacent elements. As shown in Figure 5, the system will show detailed feedback regarding the student's submission, indicating how it failed/succeed on the corresponding test cases.

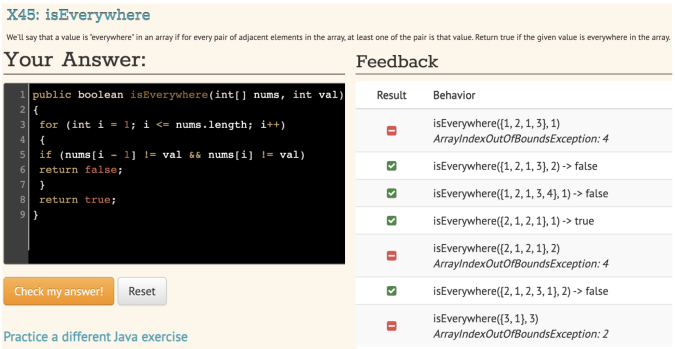


Figure 5: The CodeWorkout interface, with the problem description on the top, the coding area in the middle, and the feedback on the right.

The data collected from CodeWorkout is in Progsnap2 [33] format, and consists of two semesters: Spring 2019 and Fall

[1]https://codeworkout.cs.vt.edu/

2019. Similar to iSnap, we processed the data to eliminate factors that might affect students' problem-solving progress, and only kept the first compliable program from each student. In total, we have 448 and 307 student submissions from each semester, respectively. Please note that in CodeWorkout, only submissions from students are recorded and sequences of student edits are not available, thus it is only evaluated on the task of student program classification.

3.2 Task Description

For the task of student program classification, the ground truth labels are generated as follows: in iSnap, a student's submission is correct only if it satisfies all rubrics requirements, which are based on the expert-designed features and verified by humans; in CodeWorkout, a submission is correct if it passes all testing cases. Table 1 shows the number of correct and incorrect submissions across the semesters in each dataset. Note that here we include one submission per student to ensure that all data points are independent in both datasets. More specifically, for each student, we include the student's "own" submission before receiving any detailed feedback, which means the student's final submission for iSnap and the first submission for CodeWorkout.

Table 1: Data overview on Student Program Classification

Semester	iSnap	
	correct	incorrect
S16	24	41
F16	16	22
S17	12	17
F17	11	28
Total	63	108

Semester	CodeWorkout	
	correct	incorrect
S19	156	151
F19	223	265
Total	379	416

3.3 Experiments

3.3.1 Models Configuration

We conducted a series of experiments across both domains by comparing ASTNN against the state-of-the-art model *Code2Vec* and three *token-based* classic ML models.

Three Token-based ML Models: Three classic ML models, K-Nearest Neighbors (KNN), Logistic Regression (LG), and Support-Vector Machine (SVM) are explored. Following prior token-based approach, we applied TF-IDF to extract textual features [42, 34]. The input sentence for TF-IDF is the sequence of AST-tokens, which is generated by the pre-order traversal of original ASTs. For each of the three models, we explored different parameters to obtain the best results. For KNN, we had $k = 10$, for LG we used $L1$ regularization, and for SVM we used *linear* kernel. Those parameters are tuned from 10-fold cross-validation with grid search, and all three models are implemented through the sklearn library.

Two AST-based Deep Learning Models: Code2Vec takes a set of AST-based paths as input, where the number of paths may vary from different student submissions. Thus we manually padded the number of paths to 100 over all code submissions. During the training, we set the maximum training epochs as 200, with the *patience* of early stopping set to 100, tuned learning rate to 0.0002. Linear layer and embedding dimensions are kept default to 100. To ensure a highest efficiency of the model, we set the batch size as the full batch. For ASTNN, the inputs are a set of ST-trees, and we padded the statement sequences to the maximum length to accommodate the longest sequence before feeding to Bi-GRU. During the training, we leverage 32 as batch size, 0.001 as learning rate, and keep the max training epoch as 50. The encoding dim for the statement encoder is set to 128, and the number of hidden neurons for Bi-GRU is set to 100. We implemented both ASTNN and Code2Vec in Pytorch. Same as the classic models, 10-fold cross-validation was applied for hyperparameter tuning.

For the task of student program classification, we did not compare ASTNN and Code2Vec against any models that used expert-designed features for two reasons: one is that the expert-designed features are only available for iSnap but not CodeWorkout; and the other reason is that these expert-designed features are used to determine the ground truth label of the student's final submission in iSnap.

3.3.2 Evaluation Metrics

Our models were evaluated using Accuracy, Precision, Recall, F1 Score, and AUC (Area Under ROC curve). Accuracy represents the proportion of students whose labels were correctly identified. Precision is the proportion of students who were predicted to be *incorrect* by each model were actually in the *incorrect* group. Recall tells us what proportion of students, who will actually be *incorrect*, were correctly recognized by the model. F1 Score is the harmonic mean of Precision and Recall that sets their trade-off. AUC measures the ability of models to discriminate groups with different labels. Given the nature of the task, in the following, we consider Accuracy and AUC as the most important metrics because the former is most commonly accepted while AUC is believed to be generally more robust.

Finally, it is important to emphasize that all models were evaluated using semester-based temporal cross-validation for both domains in this task, which only applied data from previous semesters for training and is a much stricter approach than the standard cross-validation.

3.4 Results

Table 2 and 3 compare the performing of the five models in iSnap and CodeWorkout respectively. In iSnap, among the three token-based models, LG and SVM have very similar performance as both have an accuracy score of 0.6604; moreover the best AUC and Precision are from LG and the best Recall and F1 are from SVM. Both LG and SVM outperform KNN on all metrics. While in CodeWorkout, Table 3 shows that the best accuracy, AUC, and Precision are from SVM and the best Recall and F1 are from KNN. Between the two AST-based models, ASTNN outperforms Code2Vec in both domains. It suggests that across the two different student programming environments, ASTNN is more effective than Code2Vec on the task of student program classification.

The comparisons between AST-based models with token-based models show the former significantly out-perform the latter in both domains; the only exception is that SVM with token has the highest precision in Java (Table 3). Note that here the difference between the SVM and ASTNN on

Table 2: Student Program Classification Results in iSnap

Feature	Model	Accuracy	Precision	Recall	F1_score	AUC
Majority Baseline		0.6321	-	-	-	0.5
Tokens	**KNN**	0.6132	0.7321	0.6119	0.6667	0.6137
	LG	**0.6604**	**0.8298**	0.5821	0.6842	**0.6885**
	SVM	**0.6604**	0.7460	**0.7015**	**0.7231**	0.6456
ASTs	**Code2Vec**	0.6810	0.8038	0.6786	0.7239	0.7017
	ASTNN	**0.8113****	**0.8730****	**0.8209****	**0.8462****	**0.8079****

Note: best models in each group are in **bold**, and the overall best labeled with **

Table 3: Student Program Classification Results in CodeWorkout

Feature	Model	Accuracy	Precision	Recall	F1_score	AUC
Majority Baseline		0.5430	-	-	-	0.5
Tokens	**KNN**	0.8709	0.8915	**0.8679**	**0.8795**	0.8712
	LG	0.8299	0.8922	0.7811	0.8330	0.8345
	SVM	**0.8770**	**0.9437****	0.5093	0.6616	**0.8822**
ASTs	**Code2Vec**	0.9241	0.9299	0.9359	0.9329	0.9475
	ASTNN	**0.9529****	**0.9416**	**0.9736****	**0.9573****	**0.9509****

Note: best models in each group are in **bold**, and the overall best labeled with **

Precision is rather small while the former has a much worse accuracy, F1-score, and AUC than ASTNN.

To summarize, our results show that in both domains, ASTNN achieves the best performance. These results show that by capturing the meaningful *linguistic* structure in student code, ASTNN is indeed more robust on the task of student program classification. Given its effectiveness, we further explored the effectiveness of Temporal-ASTNN which combines ASTNN with powerful temporal model LSTM on the task of student success early prediction.

4. STUDENT SUCCESS EARLY PREDICTION

For student success early prediction task, Temporal-ASTNN is compared against the original ASTNN and other temporal models. As mentioned in Section 3.1.2, here we only explored the early prediction task in iSnap.

4.1 Task Description

In iSnap, we have a total of 171 students and 31,064 temporal snapshots. Following the definitions used in prior research [23], the *successful* students are those who completed the programming assignment within one hour *and* got full credit while the rest are counted as *unsuccessful*. We have 59 *successful* and 112 *unsuccessful* ones. The detailed statistics for iSnap dataset are shown in Table 4. Note that for the purpose of learning, unsuccessful students are of interest for this classification task.

To predict student early success, we are given the first *up to n minutes* of a student's sequence data and our goal is to predict whether the student will successfully complete the programming assignment at any given point in the remainder of the sequence. To conduct this task, we left-aligned all the students' trajectories by their starting times and our *observation window* (the part of data used to train and test different machine learning models) includes the sequences from the very beginning to the first n minutes. If a student's trajectory is less than n minutes, our observation window will include their entire sequence *except* the last one.

It is worth noting that student success early prediction is a much more challenging task compared to program classification: 1) besides the *linguistic* nature in student code, it also involves temporal information, and 2) the observation window is very early and thus student final submissions are not available for training or testing.

4.2 Experiments

4.2.1 Models Configuration

To further explore the power of ASTNN, we did extensive experiments and compared it with the start-of-the-art expert-designed features [43] and token-based features on the student success early prediction task. For each of the feature embedding (expert, token, AST), we explored two categories of models: the *last value*-based Logistic Regression (LG) models, and the *temporal* LSTM models. Note that LG is selected because, among the three classic ML methods explored on the task of student program classification in iSnap, LG has achieved the highest accuracy and AUC.

Last-Value Models: Motivated by prior work, we used a "Last Value" approach [4, 37, 23] to treat the last measurements within the given observation window as the input to train models. For early prediction settings, we truncated all the sequences in the training dataset in the same way as the testing dataset. For example, when our observation window is the first 4 minutes, we will only apply the last values in

Table 4: Detailed data statistics for iSnap, including total steps, total time spent in minutes, and the success labels distribution for each of the four semesters.

Semester	Total Steps				Total Time (minutes)				Success Labels	
	min	max	median	mean(std)	min	max	median	mean(std)	successful	unsuccessful
S16	10	1024	169	199 (175)	0.533	95.667	20.733	22.777 (17.149)	23	42
F16	28	884	121	167 (168)	3.283	119.083	16.325	22.379 (24.177)	15	23
S17	15	439	75	112 (94)	2.817	62.983	14.167	16.347 (11.872)	12	17
F17	10	2276	100	219 (376)	1.65	189.667	19.1	28.224 (33.869)	9	30

Table 5: iSnap Student Success Early Predictions at First-4-minute Only

Data	Feature	Model	Accuracy	Precision	Recall	F1_score	AUC
Majority Baseline			0.6604	-	-	-	0.5
Last-Value	**Expert**	**Expert-LG**	0.6226	**0.8261****	0.5429	0.6552	**0.6603**
	Tokens	**Token-LG**	0.5566	0.7170	0.5429	0.6179	0.5631
	ASTs	**ASTNN**	**0.6698**	0.7612	**0.7286**	**0.7445**	0.6421
Temporal	**Expert**	**Expert-LSTM**	**0.7075**	0.7191	0.9143	0.8050	0.6099
	Tokens	**Token-LSTM**	0.6792	0.6915	**0.9286****	0.7927	0.5615
	ASTs	**Temporal-ASTNN$_{\text{Trunc}}$**	**0.7642****	0.7711	0.9143	**0.8366****	**0.6933****
		Temporal-ASTNN$_{\text{Entire}}$	0.7453	**0.7722**	0.8714	0.8188	0.6857

Note: best models in each group are in **bold**, and the overall best labeled with **

the sequence within the first-4-minute observation window and use them as inputs for each model. More specifically, we used the expert features of the last submission within the observation window to train and test expert-LG; similarly, the tokens from the last snapshot within the observation window to train and test token-LG; and the ASTs of the last submission within the observation window for both training and testing the original ASTNN.

Temporal Models: We applied LSTM to handle the temporal sequences of student code. Here we used the *temporal* sequences in the observation window for early predictions. Specifically for a given first-n-minute observation window: we used the sequences of expert features to train and test expert-LSTM; the sequences of token features to train and test token-LSTM. For Temporal-ASTNN, we explored Temporal-ASTNN$_{\text{Trunc}}$ and Temporal-ASTNN$_{\text{Entire}}$. Both models would first convert student code sequences in the observation window into sequences of AST vectors and then feed them into LSTM. They only differ on how their AST vectors are trained: the former uses truncated sequences while the latter uses entire sequences (see Section 2.1.3).

To summarize, we analyze two main model settings: *last-value* and *temporal*, together with three different feature embeddings: *expert*, *tokens*, and *ASTs*. Thus in total we explored the effectiveness of six models.

4.2.2 Evaluation Metrics

For student success early prediction, all the models are evaluated using Accuracy, Precision, Recall, F1 Score, and AUC. Similarly to the first task, we consider Accuracy and AUC as the most important metrics, and the more stringent semester-based temporal cross-validation was carried out.

4.3 Results

We present our results of student success early prediction by first comparing the effectiveness of all six models on first-4-minute early prediction and then by exploring their average performance across different observation windows up to the first-10-minute data.

4.3.1 Results at First-4-minute Only

Table 5 shows different performance measures of all the six models at first-4-minute. In the group of *Last-Value* models, ASTNN has the best accuracy, Recall and F1 scores while the best AUC and Precision are from Expert-LG, and both of them have better performance than Token-LG. Actually, in terms of accuracy, Expert-LG and Token-LG perform worse than the simple majority baseline. This is probably either because only relying on the first-4-minute is too early or because the last snapshot of the first-4-minute does not provide enough information for these models to make effective early predictions. The fact that across the five evaluation metrics, the best performance either comes from Expert feature or comes ASTNN suggests that ASTNN is comparable to expert-designed features because of its ability of handling the *linguistic* structure of student syntactic code.

In the *Temporal* group, Temporal-ASTNN based models are the best. More specifically, both Temporal-ASTNN$_{\text{Trunc}}$ and Temporal-ASTNN$_{\text{Entire}}$ outperform Expert-LSTM and Token-LSTM on accuracy, AUC, precision and F1 scores, except that the best recall is from Token-LSTM. Between the two Temporal-ASTNN models, Temporal-ASTNN$_{\text{Trunc}}$ is generally better than Temporal-ASTNN$_{\text{Entire}}$ as it achieves higher accuracy, Recall, F1-score, and AUC. This is probably because by using the truncated training data for representation learning, Temporal-ASTNN$_{\text{Trunc}}$ is more likely to capture the temporal information that are not only predictive of student success but also more likely to be observed in the testing with only the first-4-minute data.

When further comparing temporal models with last-value models, we can see that all *temporal* models achieve better accuracy than their corresponding *last-value* models. It is reasonable since *temporal* models are able to capture the temporal information related to student success from the

Table 6: iSnap Student Success Early Predictions in First-10-minute Overall

Data	Feature	Model	Accuracy	Precision	Recall	F1_score	AUC
Majority Baseline			0.6604	-	-	-	0.5
Last-Value	Expert	Expert-LG	0.6566 (0.05)	**0.8209** (0.06)**	0.6229 (0.11)	0.7017 (0.06)	**0.6725 (0.05)**
	Tokens	Token-LG	0.5528 (0.02)	0.7072 (0.03)	0.5571 (0.07)	0.6203 (0.04)	0.5508 (0.03)
	ASTs	ASTNN	**0.6642 (0.01)**	0.7635 (0.03)	**0.7200 (0.07)**	**0.7379 (0.02)**	0.6378 (0.03)
Temporal	Expert	Expert-LSTM	0.7189 (0.03)	0.7305 (0.05)	0.9343 (0.04)	0.8145 (0.01)	0.6255 (0.07)
	Tokens	Token-LSTM	0.6887 (0.02)	0.6966 (0.03)	**0.9429** (0.04)**	0.8001 (0.01)	0.5687 (0.05)
	ASTs	Temporal-ASTNN$_{\text{Trunc}}$	0.7396 (0.02)	0.7597 (0.03)	0.8914 (0.03)	**0.8190** (0.01)**	0.6679 (0.04)
		Temporal-ASTNN$_{\text{Entire}}$	**0.7472** (0.02)**	**0.7932 (0.04)**	0.8316 (0.04)	0.8110 (0.02)	**0.6943** (0.03)**

Note: best models in each group are in **bold**, and the overall best labeled with **

temporal sequences, but such information is not available to *last-value* models.

Generally speaking, Temporal-ASTNN achieves the best performance at the first-4-minute observation window, which indicates that by combining ASTNN with LSTM, the temporal-ASTNN is able to learn the *temporal* and *linguistic* knowledge from student code sequences.

4.3.2 Results in First-10-minute Overall

Figure 6 (a) and (b) report Accuracy and AUC performance respectively for four models predicting student success: three *temporal* models and the best *last-value* model, ASTNN. For each graph, x-axis is the observation window of early prediction, here we vary the observation window from the first 2 minutes up to 10 minutes; and y-axis is the Accuracy/AUC score. As shown in Table 1, students generally take 10 to 60 minutes to complete the task and thus we took a measurement every 2 minutes for the first 10 minutes to generate the early stage predictions for each model. Table 6 show the comparison of all six models for the student success early prediction in first-10-minute observation windows, we reported the mean value and corresponding standard deviation (in parenthesis) for each evaluation metric.

Table 6 shows a similar pattern as we observed earlier in Table 5. In the group of *Last-Value* models, ASTNN outperforms Expert-LG and Token-LG. Specifically, ASTNN continues to achieve the best accuracy, Recall and F1 scores in the first 10 minutes, and Expert-LG has the best AUC and Precision scores. In the group of *temporal* models, Temporal-ASTNN based models are still the best overall, with higher scores on accuracy, AUC, Precison and F1. Additionally, Temporal-ASTNN$_{\text{Entire}}$ is shown to be slightly better than Temporal-ASTNN$_{\text{Trunc}}$ as it achieves higher accuracy, AUC and Precision.

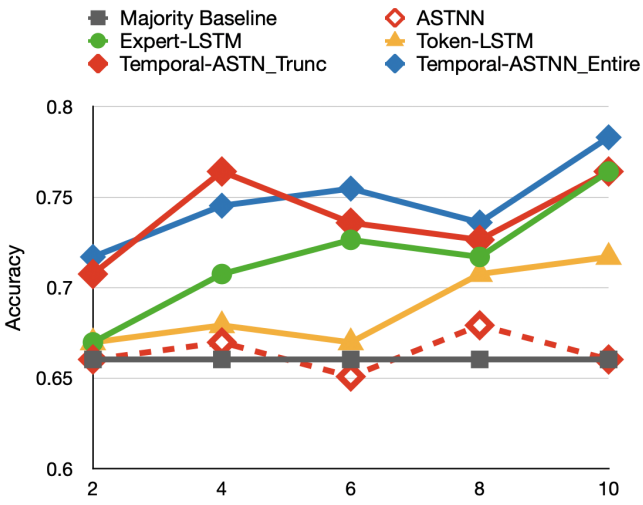
Both Figures 6 (a) and (b) show that Temporal-ASTNN$_{\text{Entire}}$ is the best model for student success early prediction as it stays on the top across all sizes of the observation window. As the length of observation window extends, all *temporal* models in general perform better, while the performance of *last-value* models fluctuates. This is because that training data includes more and more information and hereby the performance of *temporal* models improves over longer sequences. After 6 minutes, Expert-LSTM starts to perform as good as Temporal-ASTNN, which is not surprising. As the expert features are designed to detect student state for final grading, and student states will be more and more closer to their final submissions with the longer sequences. The fact that the best early predictions come from Temporal-ASTNN really suggests that addressing both *linguistic* and *temporal* nature of student code sequences brings us closer to the truth of student learning procession during programming, especially for the early stage (first 6 minutes).

5. RELATED WORK

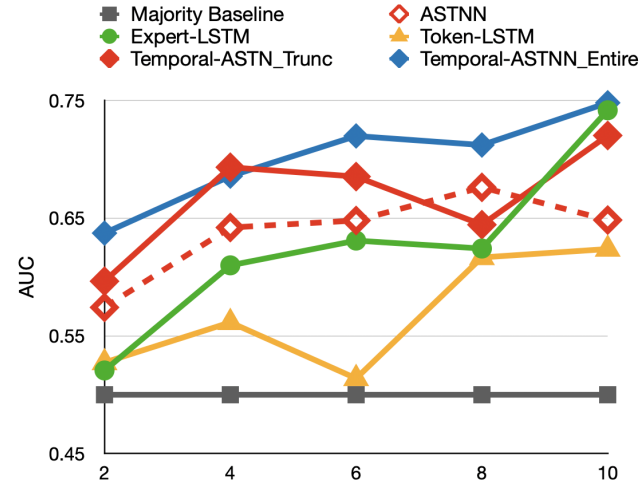
5.1 Linguistic-based Models for Programming

A wide range of work has applied NLP techniques for programming. Traditionally, some prior work directly uses the tokens of ASTs for source code tasks [38, 12], by treating programming languages as natural languages. Despite some similarities, programming languages and natural languages [25] differ in some important aspects. Programming is a complex activity, and thus programs contain rich and explicit structural information. Recently, deep learning models has shown the potential to grasp more information from AST in many tasks. For example, TBCNN [24] takes the whole AST of code as input and performs convolution computation over tree structures, and it outperforms token-based models in program functionalities classification and bubble-sort detection. In the educational domain, Piech et al. (2015) proposed NPM-RNN to simultaneously encode preconditions and postconditions into points where a program can be used as a linear mapping between these points [30]. Gupta et al. (2019) presented a tree-CNN based method, that can localize the bugs in a student program with respect to a failing test case, without running the program [14]. More recently, ASTNN and Code2Vec has shown great success.

Siting at the root of AST, ASTNN [42] was proposed to handling the long-term dependency problems when taking the large AST as input directly. AST is a form of representing abstract syntactic structure of the source code [5], and it



(a) Accuracy performance



(b) Area under ROC performance

Figure 6: Student Success Early Prediction on *iSnap*, last-value models are in dashed lines with empty symbols, temporal models are in solid lines with solid symbols, dark grey lines are from the majority baseline.

has been widely used in the domain of source code analysis. Similar to long texts in NLP, large ASTs can make deep learning models venerable to gradient vanishing problems. To address the issue, ASTNN splits the large AST of one code fragment into a set of small trees in statement-level and performs code vector embedding. It achieves state-of-the-art performance in both code functionalities classification and clone detection.

Code2Vec [1], on the other hand, utilizes AST-based paths and attention mechanism to learn code vector representation. Instead of a set of ST-trees, it takes a collection of leaf-to-leaf paths as input, and applies an attention layer to average those vectors. As a result, the attention weights can help to interpret the importance of paths. Code2Vec has shown to be very effective in predicting the names for program entities. Shi et al. (2021) also applied Code2Vec on a block-based programming dataset and used the learned embedding to cluster incorrect student submissions [34].

As far as we know, none of prior work has directly compared the effectiveness of ASTNN against Code2Vec. And in this work, we did extensive experiments across two programming domains: one is a block-based novice programming environment where the data size is relatively small; the other is a web programming platform in Java, in which more labeled data is available. Our results consistently suggest that ASTNN is able to capture more insights from student programs for correctness prediction.

5.2 Student Modeling for Programming

Student modeling has been widely and extensively explored by utilizing student temporal sequences. For example, BKT [8] and BKT-based models have been shown to be effective in predicting students' overall competence [26], predicting the students' next-step responses [41, 3, 27, 20], and the prediction of post-test scores [18, 22]. In recent years, deep learning models, especially Recurrent Neural Network (RNN) or RNN-based models such as LSTM have also been explored in student modeling [29, 36, 17, 39, 40, 19]. Some work showed that LSTM has superior performance over BKT-based models [22, 29] or Performance Factors Analysis [28]. However, it has also been shown that RNN and LSTM did not always have better performance when the simple, conventional models incorporated other parameters [17, 39].

In the programming domain, prior research has explored various temporal models for modeling student learning progression. For example, Wang et al. (2017) applied a recursive neural network similar to [30] as the embedding for student submission sequence, then feed them into a 3-layer LSTM to predict the student's future performance. Please note that the work is quite different from our proposed Temporal-ASTNN. In Temporal-ASTNN, all the components are optimized together during training, while they applied a global embedding to generate the input sequences for LSTM. On the other hand, Emerson et al. (2019) have utilized four categories of features: prior performance, hint usage, activity progress, and interface interaction to evaluate the accuracy of Logistic Regression models for multiple block-based programming activities [11]. In our earlier work, we have used the expert-designed features for a block-based programming problem to train various temporal models, then made early predictions on student learning outcomes [21, 23].

To our best knowledge, while most of the previous studies on analyzing student programming data treated student code as either *linguistic* or *temporal*, no prior work has combined the two characteristics of programming data for student learning progression. Thus our proposed Temporal-ASTNN is the first attempt to addressing *both* aspects in student code.

6. CONCLUSIONS

Tracing student learning progression at early stage is a crucial component of student modeling, since it allows tutoring systems to intervene by providing needed support, such as a hint, or by alerting an instructor. Both prediction tasks involved in this work are challenging, especially the early prediction task because: 1) the open-ended nature of programming environment hinders the prediction of student fi-

nal success, and 2) it is extremely hard to learn a meaningful representation from student code. In this work, we conducted a series of experiments to investigate the effectiveness of Temporal-ASTNN for student learning progression. We first evaluated ASTNN against Code2Vec on the task of classifying the correctness of student programs across two domains. Our results show that ASTNN consistently outperforms the other models including Code2Vec and other token-based baselines in both domains. And we can also find that AST-based models generally achieve better performance than token-based models, which is consistent with prior research [24, 42]. In the second task of student early prediction, we explored three different categories of features: expert, tokens, and ASTs. And further compared Temporal-ASTNN with other temporal models embedded with different feature set, as well as non-temporal baselines. Our findings can be concluded as follows: 1) temporal models usually outperforms non-temporal (last-value) models; 2) token-based models can only capture very limited information from student code; and 3) Temporal-ASTNN is the best out of all models in the early prediction task, it can achieve good performance with only the first-4-minute data.

Limitations: There are two main limitations in this work. First, we only explored the effectiveness of Temporal-ASTNN on one important student modeling task in one programming environment, and thus it is not clear whether the same results will hold for different tasks or in other programming domains. Second, time-aware LSTM [6] has shown to outperform LSTM on various early prediction tasks [23], while in this work we only compared our Temporal-ASTNN against normal LSTM without considering time-awareness. Nevertheless, one of the main goal in this work is to investigate the robustness of Temporal-ASTNN from both sequential and temporal embedding. Thus we have two different type of models (last-value vs. temporal) as well as another two different features (expert and tokens). Our experiments results have shown its superiority on both aspects, but still, we are not clear about the effects of time-awareness.

Future Work: An important direction for future work is to investigate the time-awareness on Temporal-ASTNN to determine how it contributes to the model in the same task. In addition, we are planning to employ Temporal-ASTNN to other temporal tasks or different domains to explore whether it continues to support improvement for programming environments. Also, this work will be applied to larger groups of students and longer programming tasks, along with integration of more informative features such as intervention and demographic features to develop more robust models.

7. ACKNOWLEDGMENTS

This research was supported by the NSF Grants: EXP: Data-Driven Support for Novice Programmers (1623470), Integrated Data-driven Technologies for Individualized Instruction in STEM Learning Environments(1726550), CAREER: Improving Adaptive Decision Making in Interactive Learning Environments (1651909), and Generalizing Data-Driven Technologies to Improve Individualized STEM Instruction by Intelligent Tutors (2013502).

8. REFERENCES

[1] U. Alon, M. Zilberstein, O. Levy, and E. Yahav. code2vec: Learning distributed representations of code. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29, 2019.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] R. S. Baker, A. T. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *ITS*, pages 406–415, 2008.

[4] I. Batal, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht. Mining recent temporal patterns for event detection in multivariate time series data. In *SIGKDD*, pages 280–288. ACM, 2012.

[5] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier. Clone detection using abstract syntax trees. In *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*, pages 368–377. IEEE, 1998.

[6] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou. Patient subtyping via time-aware lstm networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 65–74, 2017.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[8] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *UMUAI*, 4(4):253–278, 1994.

[9] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[10] S. H. Edwards and K. P. Murali. Codeworkout: short programming exercises with built-in data collection. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 188–193, 2017.

[11] A. Emerson, F. J. Rodríguez, B. Mott, A. Smith, W. Min, K. E. Boyer, C. Smith, E. Wiebe, and J. Lester. Predicting early and often: Predictive student modeling for block-based programming environments. *International Educational Data Mining Society*, 2019.

[12] J.-R. Falleri, F. Morandat, X. Blanc, M. Martinez, and M. Monperrus. Fine-grained and accurate source code differencing. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 313–324, 2014.

[13] D. Garcia, B. Harvey, and T. Barnes. The Beauty and Joy of Computing. *ACM Inroads*, 6(4):71–79, 2015.

[14] R. Gupta, A. Kanade, and S. Shevade. Neural attribution for semantic bug-localization in student programs. *Network*, 1(P2):P2, 2019.

[15] A. Hindle, E. T. Barr, M. Gabel, Z. Su, and P. Devanbu. On the naturalness of software. *Communications of the ACM*, 59(5):122–131, 2016.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing? *arXiv preprint arXiv:1604.02416*, 2016.

[18] C. Lin and M. Chi. Intervention-bkt: incorporating instructional interventions into bayesian knowledge tracing. In *ITS*, pages 208–218. Springer, 2016.

[19] C. Lin and M. Chi. A comparisons of bkt, rnn and lstm for learning gain prediction. In *AIED*, pages 536–539. Springer, 2017.

[20] C. Lin, S. Shen, and M. Chi. Incorporating student response time and tutor instructional interventions into student modeling. In *UMAP*, pages 157–161. ACM, 2016.

[21] Y. Mao. One minute is enough: Early prediction of student success and event-level difficulty during novice programming tasks. In *In: Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019)*, 2019.

[22] Y. Mao, C. Lin, and M. Chi. Deep learning vs. bayesian knowledge tracing: Student models for interventions. *JEDM*, 10(2):28–54, 2018.

[23] Y. Mao and S. Marwan. What time is it? student modeling needs to know. In *In proceedings of the 13th International Conference on Educational Data Mining*, 2020.

[24] L. Mou, G. Li, Z. Jin, L. Zhang, and T. Wang. Tbcnn: A tree-based convolutional neural network for programming language processing. *arXiv preprint arXiv:1409.5718*, 2014.

[25] J. F. Pane, B. A. Myers, et al. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, 54(2):237–264, 2001.

[26] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *UMAP*, pages 255–266. Springer, 2010.

[27] Z. A. Pardos and N. T. Heffernan. Kt-idem: Introducing item difficulty to the knowledge tracing model. In *UMAP*, pages 243–254. Springer, 2011.

[28] P. I. Pavlik, H. Cen, and K. R. Koedinger. Performance factors analysis –a new alternative to knowledge tracing. In *AIED*, pages 531–538, 2009.

[29] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *NIPS*, pages 505–513, 2015.

[30] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas. Learning program embeddings to propagate feedback on student code. In *International conference on machine Learning*, pages 1093–1102. PMLR, 2015.

[31] T. Price, R. Zhi, and T. Barnes. Evaluation of a data-driven feedback algorithm for open-ended programming. *International Educational Data Mining Society*, 2017.

[32] T. W. Price, Y. Dong, and D. Lipovac. iSnap: Towards Intelligent Tutoring in Novice Programming Environments. In *Proceedings of the ACM Technical Symposium on Computer Science Education*, pages 483–488, 2017.

[33] T. W. Price, D. Hovemeyer, K. Rivers, G. Gao, A. C. Bart, A. M. Kazerouni, B. A. Becker, A. Petersen, L. Gusukuma, S. H. Edwards, et al. Progsnap2: A flexible format for programming process data. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pages 356–362, 2020.

[34] Y. Shi, K. Shah, W. Wang, S. Marwan, P. Penmetsa, and T. Price. Toward semi-automatic misconception discovery using code embeddings. In *The 11th International Conference on Learning Analytics & Knowledge (LAK 21)*, 2021.

[35] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, 2011.

[36] S. Tang, J. C. Peterson, and Z. A. Pardos. Deep neural networks and how they apply to sequential education data. In *L@S*, pages 321–324. ACM, 2016.

[37] L. Wang, A. Sy, L. Liu, and C. Piech. Learning to represent student knowledge on programming exercises using deep learning. *International Educational Data Mining Society*, 2017.

[38] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest. Automatically finding patches using genetic programming. In *2009 IEEE 31st International Conference on Software Engineering*, pages 364–374. IEEE, 2009.

[39] K. H. Wilson, Y. Karklin, B. Han, and C. Ekanadham. Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation. *arXiv preprint arXiv:1604.02336*, 2016.

[40] X. Xiong, S. Zhao, E. Van Inwegen, and J. Beck. Going deeper with deep knowledge tracing. In *EDM*, pages 545–550, 2016.

[41] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *AIED*, pages 171–180. Springer, 2013.

[42] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu. A novel neural source code representation based on abstract syntax tree. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 783–794. IEEE, 2019.

[43] R. Zhi, T. W. Price, N. Lytle, Y. Dong, and T. Barnes. Reducing the state space of programming problems through data-driven feature detection. In *EDM (Workshops)*, 2018.

Exploring Policies for Dynamically Teaming Up Students through Log Data Simulation

Kexin Bella Yang[1], Vanessa Echeverria[2], Xuejian Wang[1],
LuEttaMae Lawrence[1], Kenneth Holstein[1], Nikol Rummel[3], and Vincent Aleven[1]

[1]Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, 15217, USA.
{kexiny, xuejianw, llawrenc}@andrew.cmu.edu, {kjholste, aleven}@cs.cmu.edu

[2] Escuela Superior Politécnica del Litoral, ESPOL, Guayaquil, Ecuador
vanechev@espol.edu.ec

[3] Ruhr-Universität Bochum Universitätsstraße 150 D - 44801 Bochum, Germany
nikol.rummel@rub.de

ABSTRACT

Constructing effective and well-balanced learning groups is important for collaborative learning. Past research explored how group formation policies affect learners' behaviors and performance. With the different classroom contexts, many group formation policies work in theory, yet their feasibility is rarely investigated in authentic class sessions. In the current work, we define *feasibility* as the ratio of students being able to find available partners that satisfy a given group formation policy. Informed by user-centered research in K-12 classrooms, we simulated pairing policies on historical data from an intelligent tutoring system (ITS), a process we refer to as *SimPairing*. As part of the process for designing a pairing orchestration tool, this study contributes insights into the feasibility of four dynamic pairing policies, and how the feasibility varies depending on parameters in the pairing policies or different classes. We found that on average, dynamically pairing students based on their in-the-moment wheel-spinning status can pair most struggling students, even with moderate constraints of restricted pairings. In addition, we found there is a trade-off between the required knowledge heterogeneity and policy feasibility. Furthermore, the feasibility of pairing policies can vary across different classes, suggesting a need for customization regarding pairing policies.

Keywords

Peer tutoring, Learning Group formation (LGF), Pairing Policies, CSCL

1. INTRODUCTION

Constructing effective, well-balanced learning groups is an important task in computer-supported collaborative learning (CSCL) [1–3]. The importance of learning group formation (LGF) has been validated empirically [4,5]. For instance, Webb et al.'s experiment proved that group composition had a major impact on the quality of group discussion and students' test scores, both during group work and subsequent individual tests [5]. The majority of existing approaches to LGF, do not support *dynamic* group formation [1]. Dynamic group formation refers to the process of groups "created on demand while various domain-specific restrictions have to be considered" [6], or can "adapt to and benefit from previous information about group members and their abilities" [7,8]. Compared to static, pre-planned LGF, the dynamic composition of groups allows for quick regrouping of learners based on up-to-date information regarding their progress and struggle. Dynamic group formation is an interesting issue, as researchers start envisioning more sophisticated and personalized classroom interactions [9] and more fluid social transitions (i.e., student social transitions that occur not all at the same time for everyone in the class) [10], that are more challenging to orchestrate.

In the context of an Intelligent Tutoring System (ITS) that supports both individual and collaborative learning, it is useful to investigate whether dynamically switching students between the two modes, as the need arises, can be effective and feasible. Pairing policies that work well in practice ideally have characteristics of both effectiveness and feasibility. By effective we mean that the pairing policy leads to students' reaching desired learning goals, and by feasible we mean that enough partners can be found under the given grouping policies (i.e., good policy coverage). Specifically, we defined *feasibility* as the percentage of students who can be teamed up under a given pairing policy.

The feasibility of LGF is an important issue to investigate in designing orchestration tools for teachers, and can be a central concern at the initial stage of tool design. This is because during the initial design stages we often do not yet have data to rigorously evaluate the effectiveness of LGF, given testing the LGF requires human resources of learners, instructors, materials resources of devices, systems, and a long time period. Additionally, an effective pairing policy that only covers a small percentage of students in a classroom may have limited influence for the whole class. Thus, the feasibility of LGF can be important in providing context for the potential coverage of LGF in a class.

Kexin Yang, Xuejian Wang, Vanessa Echeverria, Luettamae Lawrence, Kenneth Holstein, Nikol Rummel and Vincent Aleven "Exploring Policies for Dynamically Teaming Up Students through Log Data Simulation". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 183-194. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Literature on LGF in collaborative learning is vast. Researchers have paired students based on gender [11,12], learning style [13–16], students' social network [17], and their intelligence or task proficiency [18–20]. Heterogeneous and homogeneous group formation are two main approaches in team formation, and many studies have demonstrated their effectiveness in CSCL [1,8,18,20–22]. Students' knowledge level is argued to be the most suitable and important attribute to form educational groups [8]. Prior work has also used machine learning or other algorithms to incorporate multiple factors for optimizing team formation [23–26]. However, the literature on LGF provides little insight into the feasibility of these LGF policies, especially in the ITS context.

Evaluating the feasibility of LGF policies offline, prior to implementing them in real classrooms, is challenging, given a lack of readily accessible approaches. To address this problem, we adopt a process we call "*SimPairing*", to simulate pairing policies on authentic data and evaluate their feasibility. In this process, we used transaction data from several classes of students using an ITS, collected from classroom studies conducted in U.S. middle schools. We computed and analyzed how the feasibility of several LGF policies (described below) changed as each class progressed, and how the feasibility varied across different classes. Replaying historical data to simulate possible futures (e.g., Replay Enactment [27]), has been used as a method by researchers to design tools with similar data-driven, human-centered approaches [28]. Diana et al. [29], for instance, used machine learning (ridge regression) to predict students' grades based on historical data in CS education (i.e., programming). Based on these *predicted grades* and simulated *students' "helped" status,* they determined which students needed help and which may be able to provide help. They then used a network graph of code-state to search for potential peer tutors who shared a common ancestor node with the tutee. They found that grouping low-performing students together and using better model features can increase the number of students helped. Their findings suggest that using low-level log data to group and match low-performing students with a peer tutor may be an effective way to increase the amount of help given in a classroom. In contrast, we simulated different policies selected based on literature and teachers' common practice revealed in user-centered research with K-12 teachers [10,30,31], in a mathematics education context.

The current work is, to the best of our knowledge, *the first to look at dynamic pairing policies that consider students' in-the-moment wheel-spinning status.* Identifying students who are unproductively struggling, yet failing to master the skill, (i.e., wheel spinning) is a first step to getting them unstuck [32]. While there has been significant work on modeling and predicting wheel spinning [33–35], little work has been dedicated to developing interventions to get them unstuck, with a few recent exceptions [36,37]. While a typical classroom has students who are struggling on problems and those who have excelled on the same problem, the latter students' expertise is rarely utilized. Instead, often the only source of help is the instructor, who is likely unable to help all the students who need help within the time constraints of the class period [38]. Peer tutoring (i.e., pairing a struggling student with a peer tutor) could be an effective way to help get struggling students unstuck when the instructor has their hands full.

Lastly, instead of prescribing *a specific grouping criterion, our work envisions that instructors will customize pairing policies and parameters to their classroom contexts, which prior work argued to be especially helpful in the LGF process [1,8,30,39].* Amara et al. found that most of the proposed LGF solutions do not allow instructors to customize the grouping process [1]. They argued that it is less helpful to apply a grouping solution for all types of learners, and more useful to leave the choice to instructors. Instructors can then form groups according to different learning objectives, learners' needs, activity types, and customize the LGF process according to location and time [1]. Similarly, Echeverria et al. envision adaptability in an orchestration system, which "enables teachers to select the best pairing policies based on their particular goals, needs, and classroom dynamics" [30], to be helpful for different classrooms. In the current investigation, three of the four policies we studied involve an adjustable pairing threshold or parameter, which we simulated with various values.

In sum, the current work investigates the feasibility of four dynamic LGF policies derived from user research with math teachers. We investigate from three angles: overall session simulation, class-level variance, and session-level contrasting cases. This work contributes to the feasibility results of the dynamic pairing policies, recommendations for orchestration tool design, and highlights future work regarding tools supporting dynamic LGF.

2. STUDY CONTEXT

2.1 Intelligent Tutoring Systems

This study used student transaction data collected from classroom studies in U.S. middle schools (dataset link). This data logged students' interaction with an ITS called Lynnette, which offers guided practice to students in basic equation solving. ITS (also called AI-tutors) are increasingly common in K-12 classrooms to help teachers more effectively personalize instruction [40]. As shown in Figure 1, Lynnette provides step-by-step guidance, in the form of adaptive hints, correctness feedback, and error specific messages. Lynnette supports personalized mastery learning, and has been proven to improve students' equation-solving skills in several classroom studies [41–43].

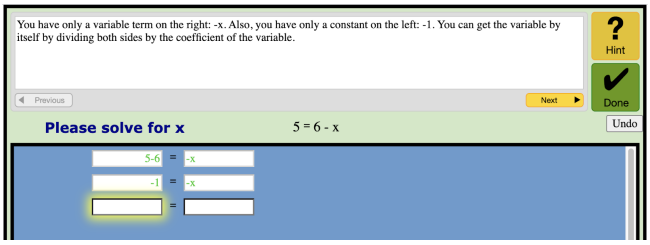


Figure 1. Example student interface for the ITS, Lynnette

The transaction data logs detailed events by the timestamp of students' interaction with the ITS, including but not limited to actions they take (e.g., requesting hints or attempting a step), knowledge components (KC) that a transaction involves, and skill mastery, calculated based on Bayesian Knowledge Tracing (BKT) student model, a two-state Hidden Markov Model. BKT is a popular student model that has been successful for various applications in the educational technology literature (e.g. [44]).

The current work lays a foundation to (in the future) use Lynnette in combination with a second ITS, APTA, which extends Lynnette's functionality to support reciprocal peer tutoring. APTA allows two students to respectively take the role of tutor and tutee. In APTA, the tutee can seek help from their partner, while the tutor can see the tutee's progress, and help them to make progress with the math problem at hand. APTA supports the peer tutor in

tutoring the tutee. Classroom studies with APTA have demonstrated that its adaptive support can improve the quality of help peer tutors give and improve students' domain learning [45,46]. A future effort for the current work is to implement feasible student pairing policies in an orchestration tool, to support teachers in dynamically pairing students to work collaboratively in APTA. Such a tool plays a key role in our vision for the smart classroom of the future, in which students alternate fluidly between individual and collaborative learning.

2.2 Wheel-Spinning Detector

Detectors have been developed to detect student behaviors of interest (e.g., gaming the system, struggling) from the transaction data. Such detectors have been used to design dashboards or tools that can alert teachers of certain student status (e.g., [9]). In our policies 1 and 2, we paired students based on their struggle status indicated by a wheel-spinning detector.

Wheel-spinning, as defined by Beck and Gong, denotes students who are failing to master a specific skill after many attempts in an intelligent tutoring system [32]. We utilized a detector that adopted the same criterion as defined by Beck and Gong [32]. The detector is embedded in LearnSphere, (i.e., a large learning analytics infrastructure) [47]. The detector considers students who have over ten practice opportunities yet still failing to reach a skill mastery on a specific knowledge component (KC) of above 0.95, to be wheel-spinning on this KC [9]. Such prolonged repeated struggles are likely to be an inefficient use of time for students [32] and may contribute to a lack of motivation for future learning [36]. Wheel-spinning is one type of unproductive struggle, and we use *struggling* and *wheel-spinning* interchangeably in this paper.

3. METHODS

We evaluated how feasible four pairing policies (described below) are, based on simulation with historical transaction data from Lynnette. We applied each pairing policy to data from each class session. For every minute in a session, we calculated the percentage of students who met the policy's criterion for being teamed up. Based on this calculation, we evaluated policy feasibility using two measures, FI_1 and FI_2, defined in 3.2. In the simulation process, we did not make assumptions about how long simulated collaboration episodes would last. We foresee that in any of these episodes, students will be given the task of collaboratively solving several math problems; it is hard to predict how long that will take them. We thus did not simulate taking tutors or tutees out of the pool of students available for teaming up, or returning them to this pool, at the beginning and end of collaborative episodes, respectively. Although this simplification might introduce some inaccuracy into the simulation results, it may be hard to do better. As well, the asymmetric roles that paired-up students have in the pairing policies may limit the inaccuracy. For example, simultaneously keeping a struggling student and a non-struggling in the pool instead of taking them both out might have offsetting effects in terms of feasibility.

Our simulation involved four pairing policies, namely:

Policy 1 - Struggle with Non-Struggle: Pairing students who are wheel-spinning (unproductive struggle) with students who are not wheel-spinning.

Policy 2 - Pairing with Restriction: Pairing students who are wheel-spinning with those who are not wheel-spinning, with a varying pairing restriction (PR) rate β. The PR rate simulates restrictions regarding who can collaborate with whom, which in real life would be provided by the teacher.

Policy 3 - Knowledge Difference Pairing: Pairing students whose knowledge levels (as measured by the tutor's BKT) differ by *more* than a certain threshold α.

Policy 4 - Knowledge Similarity Pairing: Pairing students whose knowledge levels (as measured by the tutor's BKT) differ by *less* than a certain ceiling γ.

The distinction in these policies aligns with Amara et al.'s categorization for dynamic group formation [1]: *intra-session* and *inter-session* grouping. Intra-session grouping allows for changing group members during the learning process, which is useful, for example, for synchronous mobile collaborative learning [1]. In *inter-session grouping,* groups are formed only before starting or after ending the learning process. Specifically, policies 1 and 2 fall under *intra-session* grouping since we simulated pairing students based on their in-the-moment struggle. These two policies also concern *fluid social transitions [10]*, since the students in a given class may transition from individual to collaborative learning at different times. Our pairing policies 3 and 4 concern inter-session grouping, and pair students based on their initial knowledge level. To apply these policies, teachers or the tutoring system would assess students' knowledge level, prior to (or at the beginning of) a class session.

The research questions we aim to answer are:

RQ1: Based on a pairing simulation done with students' historical transaction data, how feasible are the four pairing policies?

RQ2: How does varying the parameters in the pairing policies affect the feasibility of pairing students?

RQ3: Does the feasibility of the pairing policies vary for different classes or sessions, if so, how?

3.1 The Four Pairing Policies

3.1.1 Policy 1: Struggle with Non-Struggle

Description. Policy 1 utilizes the struggle detector (section 2.2) to pair students who are wheel-spinning with those who are not. The struggle detector assumes students' wheel-spinning status to be a binary value for a given timestamp. Inspired by the work of Diana et al. [29], we categorized students in the *Struggle Pool* if they were wheel-spinning on at least one KC, indicating they could need help from a partner. Students not wheel-spinning on *any* KC were categorized in the *Tutor Pool* and considered as available tutors. We simulated pairing students in the *Struggle Pool* with students in the *Tutor Pool*. To determine the feasibility of this policy, we calculated the percentage of struggling students who had a potential partner (for more detail, see below).

Rationale. Literature suggests that when students are wheel-spinning, giving them more of the same type of math problems to solve may not be productive [36]. When wheel-spinning, students would likely benefit from instructor attention or extra instruction. However, prior user research in the classroom (e.g. [9]) found that teachers often cannot help all struggling students. In this case, wheel-spinning students may benefit from a peer tutor's help, which leads to a policy that seeks to dynamically find them partners [36].

3.1.2 Policy 2: Pairing with Restriction

Description. Policy 2 is an extension to Policy 1, where we pair a struggling student with a non-struggling student, while enforcing a constraint that not all students are eligible for teaming up. The proportion of ineligible students is captured as the Pairing

Restriction (PR) rate. The PR rate is used to simulate situations where the teacher prefers that certain students do not work together. Specifically, we simulate pairing students in the *Struggle Pool* with students in the *Tutor Pool,* while enforcing the restriction that β% ($0 < \beta < 1$, step = 0.1) of students in the *Tutor Pool* are ineligible as partners. For example, a PR rate β of 0.2 means 20% of the students in *Tutor Pool* have been restricted from working with any students in *Struggle Pool*. It is important to know how these restrictions affect the feasibility of the policies.

Rationale. We designed this policy based on results found in a survey we conducted with 54 middle-school math teachers on their pairing preferences in collaborative learning [31] and semi-structured interviews conducted with middle school teachers. Teachers expressed a desire to set constraints so that certain pairs of students are restricted from working together. Previous studies and user research by Olsen et al. and Echeverria et al. also informed the idea of ruling out certain pairings in advance [10,30]. Such restrictions usually arise from information or concerns teachers have about their students' traits, behaviors and interpersonal relationships [8].

3.1.3 Policy 3: Knowledge Difference Pairing

Description. In Policy 3, we pair students who have *different* Initial Knowledge (IK) levels. In a practical scenario, teachers may assess students' knowledge through quizzes or exams. Alternatively, if the classrooms use ITS, teachers may have students practice several math questions individually, prior to transitioning into collaborative learning activities.

To simulate this policy without having pre-assessment data, we used data from the tutoring sessions (captured in the log data) to compute students' IK levels. Specifically, we computed a student's IK for each KC, as the average mastery for their first *three* opportunities for this KC. The reason is we want to use up only a small portion of the data from the tutoring session, so the measure represents initial knowledge. In our datasets, three opportunities generally fall in the first quartile (25%) of students' total number of opportunities for any given KC. Another reason we chose the cutoff of three is a previous EDM study with ASSISTments data showed student learning often appeared to occur, after students have had *ten* opportunities with the target knowledge [48]. Thus one may assume learners to have little learning on their first three times in transaction data practicing a KC. A student's overall IK S_j $(j \in N)$ is calculated as the average of their IK across KCs. To more accurately calculate students' *IK*, we limit our simulation to sessions that practiced the first (i.e., the most basic) level of KCs, involving 25 sessions.

KD was the difference between two students' IK, and denoted as S_{jk} $(j,k \in N)$, which was calculated as the absolute value of differences between two students' IK:

$$KD(S_{jk}) = |IK(S_j) - IK(S_k)|$$

Inspired by Huang and Wu's work that proposed a clustering LGF method that considers a threshold of learner heterogeneity [49], this work similarly considers a KD threshold. For this policy, the required KD of two students (S1, S2) should be a *minimum* of α ($0 < \alpha < 1$, step = 0.1) for them to be eligible to pair up.

Rationale. The heterogenous pairing policy was informed by findings from user research with math teachers. In the survey conducted with 54 math teachers, we found the most common way teachers paired students was pairing those who have a different level of knowledge (67%, N = 34) [31]. In our study, we use students' *mastery of knowledge components* (i.e., targeted math skills) calculated based on the BKT model to represent students' knowledge. In a systematic literature review on LGF in CSCL, Maqtary et al. found the knowledge level is the most commonly used attribute in LGF, which they claim to be the most suitable and important attribute to form educational groups because of its effects on the group process [8].

There is a range of research that shows heterogeneous grouping can promote positive interdependence, better group performance, and effective interactions [1,49–52]. Heterogeneous group composition not only enhances elaborative thinking, but also leads learners to deeper understanding, better reasoning abilities, and accuracy in long-term retention [49,50]. Research also suggests that collaborative learning with heterogeneous group composition by characteristics such as gender, ability, achievement, social-economic status (SES), or race, can be beneficial [51].

3.1.4 Policy 4: Knowledge Similarity Pairing

Description. Policy 4 is analogous to Policy 3, with the same definition of KD and IK as in Section 3.1.3. To pair students with *similar* knowledge, using the same calculation as Policy 3, this policy simulated pairing students that have *a small* KD. To be eligible for students to form a pair under this policy, the KD of two students (S1, S2) should be *less than or equal to* γ ($0 < \gamma < 1$, step = 0.1). For example, when γ = 0.2, two students with knowledge of 0.6 and 0.75 (KD = 0.15, below γ) would be eligible to pair, but another pair with knowledge of respectively 0.5 and 0.8 (KD = 0.3, above γ) would not be eligible.

Rationale. Policy 4 was inspired by prior literature and informed by user research. Literature suggests that homogenous groups can be beneficial for students' learning. For example, Fuchs et al. found homogenous dyads generated greater cognitive conflict and produced better quality work than heterogeneous groups [22]. Additionally, among 54 teachers we surveyed, 43% reported that they pair students with a similar level of knowledge [31]. This was the third most popular grouping method that teachers commonly adopt (43%, N = 23), following strategies of pairing students with different knowledge (Policy 3) and pairing students randomly *[31]*.

3.2 Metrics

In this section, we describe the metrics to evaluate the pairing policies. We discuss how prior work informed the metric definitions, and how different metrics could be suitable to evaluate different policies. We build on Diana et al.'s work [29], who defined an Efficiency Index (EI) as a measure of a pairing algorithm's performance, specifically:

$$EI = \frac{LowPerformingStudentsHelped/BeingHelped}{LowPerformingStudents}$$

We adapted EI into two metrics of interest for our pairing policies: Feasibility Index 1 and 2. FI_1 is the percentage of students who can be paired among all struggling students in a session.

$$Feasibility\ Index\ -\ 1\ (FI_1) = \frac{StrugglingStudentsCouldBeHelped}{TotalStrugglingStudents}$$

FI_2 is the ratio of paired students among all the students in a session.

$$Feasibility\ Index\ -\ 2\ (FI_2) = \frac{StudentsPaired}{TotalStudents}$$

For Policies 1 and 2: Given the goal to pair all struggling students in the session, FI_1 was a suitable measure for policy feasibility,

showing what percentage of students who are wheel-spinning can get help. *For Policies 3 and 4:* Given the goal to pair all students in the session who satisfied a certain KD, FI_2 was a suitable measure for policy feasibility, as it calculated the percentage of the paired students out of the total students.

3.3 SimPairing Approach

There are three main steps in *SimPairing*: 1) data cleaning and preprocessing, 2) policy simulation, and 3) policy evaluation. The data cleaning and preprocessing step consists of clustering student transaction data into meaningful class sessions based on meta-data (e.g., student transaction timestamp, classes), and examining the distribution of students per class session to detect outliers. The policy simulation step takes the preprocessed transactional data and applies a pairing policy to class sessions. In the policy evaluation step, we computed the policy feasibility based on the simulation results, using the corresponding feasibility index (FI_1 or FI_2). We also observed how the FI changed by varying the parameters (i.e., KD, and PR rate).

4. ANALYSIS AND RESULTS

4.1 Data Cleaning and Preprocessing

We first clustered student transaction data into meaningful class sessions, based on timestamp, student ID, and class. We visualized student engagement for all class sessions based on transaction data, which allowed us to ensure that the sessions we analyzed had a continuous student interaction with the system, and helped us check for outliers (e.g., unusually short sessions). We excluded four outlier sessions: 2 sessions that had only 1 student, 2 sessions that lasted less than 15 min, as sessions commonly lasted 40 minutes or more.

Transaction data of a total of 68 sessions, from six middle school math classes, collected from 2013 to 2014 were used for policy simulation. It consists of 894 students and 197,234 rows of transactions. The average number of students in a session was 13 (*Min* = 5, *Max* = 24, *SD* = 25.3); the average duration of class session was 41.9 minutes (*Min* = 10, *Max* = 81, *SD* = 9.42); the average number of sessions in a class was 11 (*Min* = 3, *Max* = 23, *SD* = 9.33).

4.2 Overall SimPairing Analysis

In this section, we present, for each policy, the SimPairing analysis and the results. The goal for this analysis was to evaluate the overall feasibility of the four pairing policies (RQ1) and see how the feasibility depends on policy parameters (RQ2).

4.2.1 Policy 1: Struggle with Non-Struggle

We simulated Policy 1 for every minute in a given class session, which returned the number of struggling students who did or did not have a potential partner. Based on this we calculated the FI_1 for every minute in a class session. We then averaged FI_1 across the length of each class session, to obtain an average FI_1 for a given session. We refer to it as the Average Number of Struggling Students (ANSS). We then took the average of the ANSS across all sessions, to obtain an overall simulation result for all 68 sessions. Figure 2 (green area) shows the average FI_1 for all sessions was 0.94 (*SD* = 0.007). Thus, on average, across time, 94% of struggling students could be paired with a partner who was not struggling.

4.2.2 Policy 2: Pairing with Restriction

The Policy 2 simulation process is similar to Policy 1, with the addition of enforcing a varying PR rate. PR rate specifies a percentage of students in *Tutor Pool* as restricted from partnering with students in the *Struggle Pool*. We computed FI_1 with varying PR rates. As shown in Figure 2 (white area), FI_1 dropped as the PR rate increased, as expected. However, even with a relatively high PR rate of, for example, 0.4, meaning, 40% of non-struggling students are restricted from working with struggling students, we still get a high average FI_1 of around 0.80, (i.e., 80% of struggling students could be paired). The simulation result means that teachers can afford to set moderate restrictions for pairings, without compromising too much of the pairing policy's feasibility.

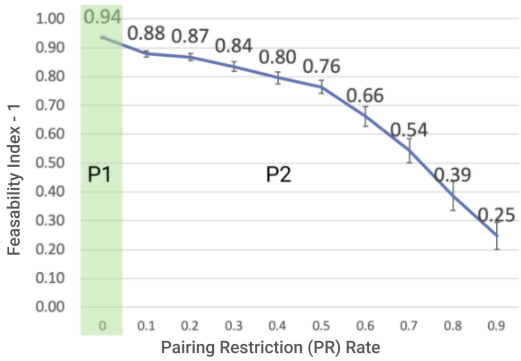


Figure 2. FI_1 for Policies 1 and 2

4.2.3 Policy 3: Knowledge Difference Pairing

Policy 3 requires students to be above a given *minimum* distance in their IK to be eligible for pairing up. We simulated this policy by computing FI_2 with varying values for the KD distance threshold α. We simulated these sessions to calculate the FI_2. As in Figure 3 (blue line), FI_2 dropped rather quickly as the required knowledge distance threshold went up. For example, the simulation results show that if we want to ensure an average of 80% of paired ratio, the KD threshold should be set to less than approximately 0.1 (i.e., a very strict bar).

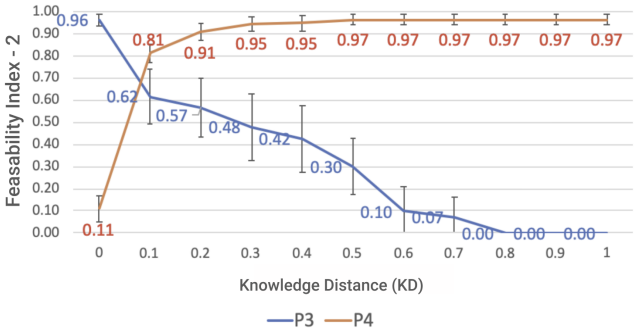


Figure 3. FI_2 for Policies 3 and 4

4.2.4 Policy 4: Knowledge Similarity Pairing

Policy 4 requires students to be below a *maximum* distance in their IK to be eligible for being paired up. We simulated this policy and computed FI_2 with different values for the KD distance ceiling γ. We found that this policy would work well even with a low, strict ceiling for the knowledge distance (Figure 3, red line). For example, when γ was 0.1, (i.e., two students' knowledge distance can be at most 0.1 for them to be teamed up), the average FI_2 was still 0.81 (*SD* = 0.08) across the class sessions involved. When γ was set to above 0.3, 95% of students in class could find an eligible partner.

4.3 Class-Level Variance Analysis

We explored how the four pairing policies worked for different classes and whether the pairing policies should be adapted to class-level differences (RQ3).

4.3.1 Class-level Differences

Based on Echeverria et al.'s insight that pairing support for teachers should ideally be adaptable to different classroom contexts [30], we analyzed, first, if there were systematic differences between different classroom contexts, and second, if these differences relate to policy feasibility differences. The main context variables taken into account by our pairing policies are *students' struggle status* (Policies 1 and 2) and *initial knowledge* (Policies 3 and 4). We thus analyzed if the classes had different *struggle statuses* and *initial knowledge (IK)*.

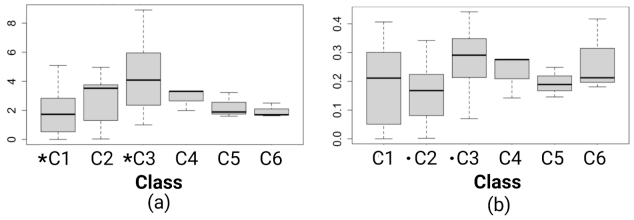


Figure 4. ANSS (a) and ASSRs (b) for Six Classes

***Student struggle status*:** We first calculated the number of students in wheel-spinning status for every minute within each class session. We then computed ANSS (section 4.2.1), and the *Average struggling students ratio (ASSR)= ANSS / total number of students in the session*. Histograms for ANSS and ASSR for all sessions show they follow the normal distribution. We conducted one-way ANOVAs, respectively taking the *ANSS* and *ASSR* as outcome variables and *Class* as the explanatory variable. The results showed a significant difference for ANSS among classes (Figure 4, a) [$F(5,62) = 4.34$, $p < 0.001$]. *Post hoc* Tukey tests showed C3 and C1 have significant differences (*diff* = 2.55, $p < 0.001$). All *post hoc* pairwise tests conducted in this study were corrected for multiple comparisons. The ANOVA result indicated that the classes differed with marginal significance [$F(5,62) = 1.94$, $p < 0.1$] (Figure 4, b). *Post hoc* Tukey tests showed a marginal difference in the ASRR between C3 and C2 (*diff* = 0.10, $p = 0.08$). Thus, there were class-level differences with respect to students' struggle status.

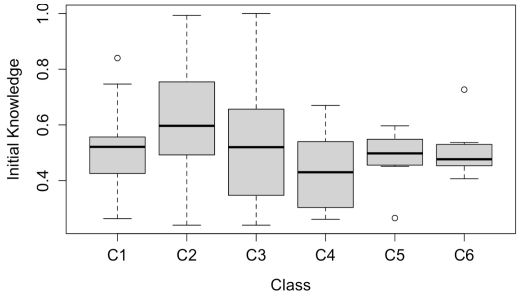


Figure 5. Initial Knowledge for Six Classes

Initial Knowledge: We calculated each student's IK for all KCs involved (defined in section 3.1.3). The histogram for all students' IK shows it follows the normal distribution. We then conducted one-way ANOVAs using *IK* as the outcome variable, and *Class* as the categorical explanatory variable. The results indicated a significant effect of classes on IK for the six classes [$F(5, 320) = 5.895$, $p < 0.05$], and the IK for the six classes were not all equal. From the *post hoc* Tukey tests comparing knowledge level between each pair of the classes, we saw significant differences between classes C2 and C1 (*diff* = 0.12, $p < 0.05$), C3 and C2 (*diff* = -0.095, $p < 0.05$), and C4 and C2 (*diff* = -0.189, $p < 0.05$). C2 had the highest median of student IK (Figure 5), and a significantly higher level of IK than C1 and C3, and C4.

Having characterized struggle and IK at a class level, we compare the policies' feasibility across classes.

4.3.2 Policies 1 and 2

Policy 1 had an average FI_1 above 0.85 (Figure 6, green area). We statistically compare if Policy 1 behaved differently for each class and see whether this policy should be adaptable for each class. Using session as the unit of analysis, we conducted a one-way ANOVA using the FI_1 for each session as the outcome variable, and *Class* as the categorical explanatory variable. The results indicated that there was not a significant effect of class on FI_1 [$F(5,62) = 1.24$, $p = 0.30$]. This result showed that Policy 1 was relatively consistent across the six classes, suggesting that Policy 1 may not need to be adaptable to classes.

For Policy 2, with increasing PR rate, the FI_1 decreases at a different speed for different classes, indicating some degree of class-level difference (Figure 6, white area). We conducted ANCOVAs with *Class* being the categorical explanatory variable, the *PR rate* as the quantitative explanatory variable, and FI_1 being the quantitative outcome variable. We first compared the model with and without a Class × PR rate interaction term. The model comparison result showed no evidence of an interaction effect among explanatory variables ($F = 1.63$, $p = 0.15$). We thus perform ANCOVA using an additive model. Results indicated there were eight pairs of classes that had significant differences in FI_1 for this pairing policy ($p < 0.05$). The eight pairs were C1-C2, C1-C3, C1-C6, C2-C3, C2-C6, C3-C5, C4-C6, and C5-C6.

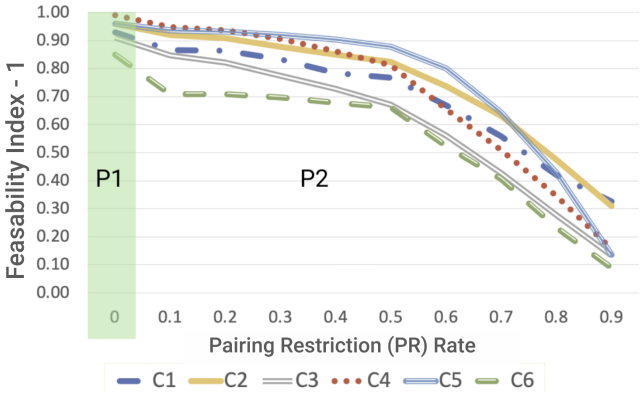


Figure 6. FI_1 of Policies 1 and 2 for Six Classes

Next, we looked at possible relations between the class-level feasibility variance of policies 1 and 2, and the class-level differences in *struggle status* (section 4.3.1). We found that for classes that differed with respect to the number of struggling students (C3-C2) and the average ratio of student struggle (C3-C1), the feasibility of Policy 2 tended to differ as well. This finding suggests that 1) Policy 2 may benefit from being adaptable to class-level characteristics, and 2) variables characterizing a class's struggle status (e.g., ANSS and ASSR) may have value in indicating how Policy 2 should be adaptable. On the other hand, the feasibility of Policy 2 was different in Class 6 compared to all other classes except C3, yet Class 6 did not differ in number or

ratio of struggle students from other classes. Thus, students' struggle status alone may not provide enough information to fully decide whether and how P2 should be adaptable.

4.3.3 *Policies 3 and 4*

For Policy 3, the classes shared a downward trend in FI_2 with different slopes for each class (Figure 7). For example, when the KD was 0.1, we saw the FI_2 values for class 6 (green dotted-line) drop to as low as 50%, but the other five classes have FI_2 above 75%. This shows that policy feasibility may be differently affected by the knowledge heterogeneity threshold in each class.

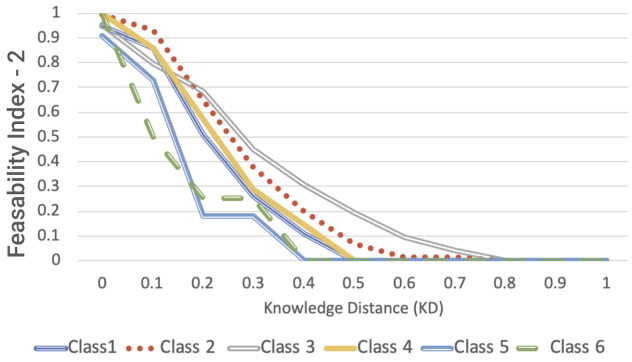


Figure 7. FI_2 of Pairing Policy 3 for Six Different Classes

To test whether Policy 3 behaved differently for each class, we conducted ANCOVAs with *Class* as a categorical explanatory variable, *KD* as a quantitative explanatory variable, and FI_2 in each session as the outcome variable. We first compared the model with and without a Class × KD interaction term. Results indicated no evidence supporting the interaction effect ($F = 0.81$, $p = 0.54$). We performed an ANCOVA using an additive model. Results indicated that three pairs of classes had significant differences in FI_2 ($p < 0.05$), and that two pairs of classes were marginally different ($p < 0.1$). They were C1-C3, C3-C5, C3-C6 ($p < 0.05$) and C2-C5, C2-C6 ($p < 0.1$) .

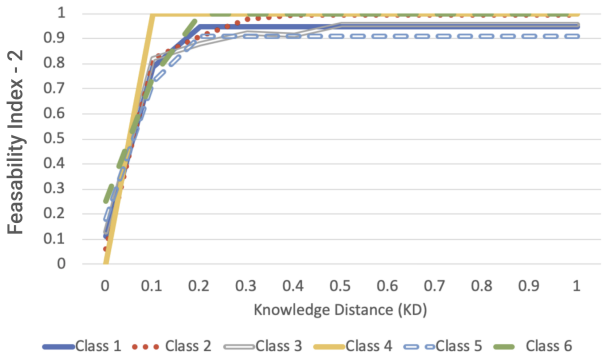


Figure 8. FI_2 of Policy 4 for Six Different Classes

For Policy 4 (Figure 8), the six classes were more convergent and clustered closer together than Policy 3 (Figure 7). This indicated the class level difference may not be as strong as that in Policy 3, which our ANCOVA tests confirmed. Similar to Policy 3, we compared whether Policy 4 behaved differently for each class. We conducted an ANCOVA, with *Class* as a categorical explanatory variable, *KD* as a quantitative explanatory variable, and FI_2 as the outcome variable. We first compared the model with and without a Class × KD interaction term. No evidence supporting interaction effect among explanatory variables ($F = 0.13$, $p = 0.99$). We then performed ANCOVA using an additive model. Results indicated that there were no significant differences in FI_2 ($p > 0.05$) for Policy 4. We confirmed a smaller class-level difference as compared to Policy 3, in KD's effect on policy feasibility. From this result, we conclude Policy 4 performed quite consistently across classes, and no significant evidence showed that Policy 4 should be adaptable to classes.

Analogous to policies 1 and 2, we then looked at relations between feasibility variance for policies 3 and 4 and class-level IK characteristics in Section 4.3.1. We observed significant differences in IK between C2-C1, C3-C2, and C4-C2. However, the differences in IK for two classes cannot accurately predict whether they had different feasibility in Policy 3 and Policy 4, and other classroom characteristics may be needed to accurately represent the class-variance of feasibility.

4.4 Analysis of Contrasting Cases

We conducted a case study to understand how policies may perform dynamically (e.g., across every minute during class time) and differently in different class sessions (RQ3). For every policy, we selected a *typical* case and an *extreme* case in terms of the policy feasibility simulation results. For the *typical* case for all four policies, we selected a session (Session 1, C1) that had an average length of time (i.e., 41 minutes), an average number of students (i.e. 13 students). In the session, policies performed typically (as by visually comparing the simulation results of each policy for all sessions). As for the *extreme case*, we examined the simulation results for each policy on each session, and identified different sessions where each policy performed surprisingly or differently from the common trend. The extreme case can be a worst case scenario (Policies 1, 2 and 3) or a case that works surprisingly well (Policy 4). Below, we present the analysis and results for these contrasting cases for each policy.

4.4.1 *Policy 1*

In Policy 1, we chose the extreme case (Session 19, C3) as it was a session that this policy has the worst performance on, and thus it had the most different FI_1 trend, from examining visualizations of FI_1 for all sessions involved. We compare the typical case and extreme case by first contextualizing the struggle status of the two cases, and comparing the visualization of feasibility (for each minute) in the two sessions. Figure 9 depicts the *ratio of struggling students* (among all students in the class session) for the contrasting cases. For Policy 1 simulation (Figure 10), we obtained, for every minute in the class session, three values regarding policy feasibility: the number of students who were not wheel-spinning on any KCs (green bar), the number of students who were struggling, and *had* a potential partner (yellow bar), and the number of students who were struggling and *did not have* a potential partner (red bar).

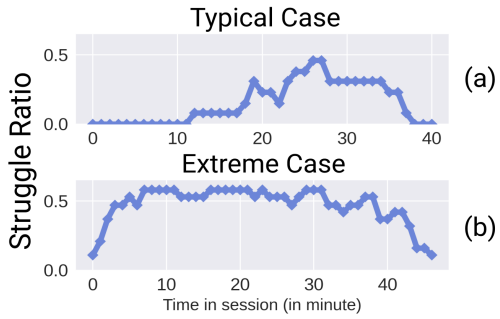


Figure 9. Struggle Ratio of a Typical (a) and Extreme (b) Case

Typical Case. In the typical case, students started to struggle after 10 minutes, as shown in Figure 9 (a) and Figure 10 (a). In all instances of wheel-spinning, a potential partner was available (i.e., $FI_1 = 1$ for any minute in this session). The typical case aligned with the overall simulation results from Policy 1, which showed that, for an average class, most struggling students could find potential partners, the minute they struggled.

Extreme Case. Among all sessions in our dataset, the per-minute struggle ratio rarely goes over 50%. By contrast, the extreme case session had more struggling students than non-struggling students in 27 out of 46 minutes, indicated by a struggle ratio of above 0.5, as shown in Figure 9 (b). This resulted in lower feasibility for Policy 1. The extreme case differs from the typical case in two aspects. First, unlike the typical case, almost as soon as the class began, students started wheel-spinning. Second, there were wheel-spinning students without potential partners in almost every minute of the session (indicated by red bars in Figure 10 (b)).

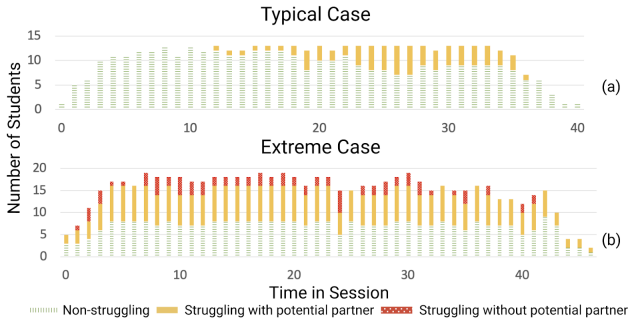


Figure 10. Policy 1 for a Typical (a) and an Extreme Case (b)

4.4.2 Policy 2

Same as in Policy 1, we chose this extreme case (Session 19, C3) as this policy had the worst performance on this session. This session also had the most different FI_1 trend. We simulated Policy 2 and calculated FI_1 for every minute in the two contrasting class sessions. Figure 11 showed the typical and extreme case, of how FI_1 changed when different PR rates were simulated. We plotted four different PR rates in the figure.

Typical Case. We saw two patterns in the Policy 2 simulation for the typical case. Firstly, the policy was typically robust in maintaining high feasibility with a non-zero (albeit low) PR rate. In Figure 11(a), lines with PR rate 0.1 and PR rate 0 completely overlapped. With these PR rates, there were no instances of struggle without a potential partner (i.e., feasibility was 1 across the whole session). Secondly, when the PR rate was high (0.5 or 0.8), FI_1 exhibited a sharp decrease, when there was an increase in student struggle. For instance, in Figure 9 (a) at minute 19, the struggle ratio increased from 0.07 to 0.23, as the number of wheel-spinning students went from 1 to 3. In Figure 11 (a) at the same time (t = 19 min), we saw a sharp decrease in FI_1 when the PR rate was 0.8.

Extreme Case. As shown in Figure 11 (b), the extreme case exhibited very different patterns compared to the typical case, mainly in three aspects. First, given it had a higher struggle ratio, even when there was no pairing restriction (i.e., PR rate = 0), we observed the FI_1 was not always 1 or even close to 1, as we saw in the typical case. Second, even a slight PR rate of 0.1 further worsened the policy feasibility and lowered the FI_1, unlike the typical case which showed resistance to a low PR rate. Third, if a class had a higher struggle ratio, the PR rate had a stronger effect on worsening FI, than for a session that had a lower struggle ratio. This effect was especially prominent when the PR rate was high (e.g., 0.5 or 0.8). This contrast means that the instructors may afford to set a higher PR rate without affecting the FI_1 too much, for a common session that has a moderate struggle ratio. However, the instructors may need to consider lowering the PR rate for a high-struggle session.

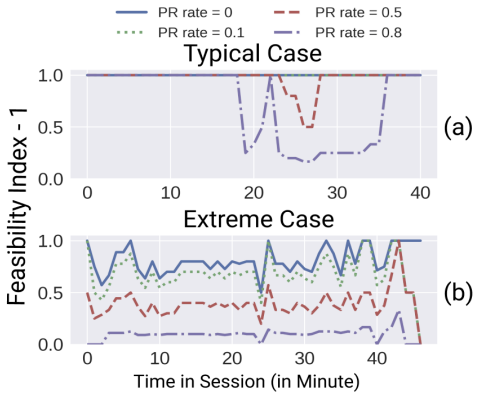


Figure 11. Policy 2 for a Typical (a) and Extreme (b) Case

4.4.3 Policy 3

In Figure 12 we present the results for Policy 3 simulation on two contrasting cases, plotting FI_2 for every step of the knowledge distance threshold for that session. The extreme case was chosen for having the most different FI_2 trend, from examining visualizations of FI_2 for all sessions involved.

Typical Case. As shown in Figure 12 (a), for the typical case, the FI_2 dropped gradually as the required KD threshold increased, which aligned with the overall simulation result. To pair students based on different knowledge (Policy 3), the instructors need to balance the required heterogeneity (i.e., higher knowledge distance threshold) and the desired paired ratio of the whole class. In this typical case, if a teacher selects a threshold of 0.5 or higher, none (0%) of students in the class session would be paired.

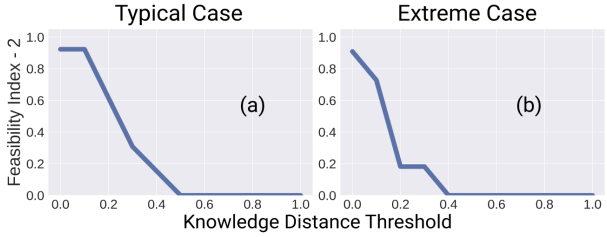


Figure 12. Policy 3 for a Typical (a) and Extreme (b) Case

Extreme Case. As shown in Figure 12 (b), for the extreme case (Session 1, C5), while the downward trend was similar, we observed a more rapid decrease as compared to the typical case. Specifically, the FI_2 dropped to only 20% when the KD threshold was as low as 0.2, compared to 60% of FI_2 at the same KD threshold in the typical case. This comparison indicated that some class sessions were more heavily influenced by the parameter of the required knowledge distance threshold, and the effect may differ from session to session.

4.4.4 Policy 4

From the previous analyses, we noted that Policy 4 performed reliably and similarly across classes, making it harder to select an

extreme case or a worst case scenario. We selected a session where Policy 4 performed surprisingly well (Session 2, C3). In Figure 13 we visualized Policy 4 simulation on two contrasting cases, plotting FI_2 for every step of the knowledge distance ceiling γ for that session.

Typical Case. The tradeoff between knowledge homogeneity and policy feasibility was less prominent than under Policy 3. This means that instructors can afford to choose a stricter (i.e., lower) ceiling so students have a very small knowledge distance, and still achieve high feasibility (FI_2). For example, in Figure 12 (a), we saw that even if the instructor chooses a very strict threshold of $\gamma = 0.1$, nearly 95% of students were able to find a potential partner.

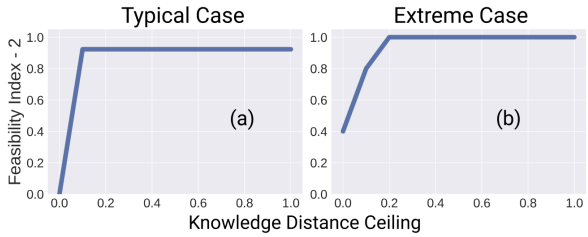


Figure 13. Policy 4 for a Typical (a) and Extreme (b) Case

Extreme Case. In Figure 13 (b), even when the KD ceiling was set to 0 (which means students must have the same level of mastery to be paired up), 40% of the students can still be paired, unlike typical cases where usually no two students have the exact same IK. Another noteworthy distinction is, while the typical case did not reach $FI_2 = 1$ with any ceiling of KD, the extreme case successfully paired all students ($FI_2 = 1$) with a relatively low ceiling of 0.2.

5. DISCUSSION

In line with previous LGF research [8], this work introduces four dynamic LGF policies contextualized in ITS and grounded in user research with K-12 teachers [10,30]. In this section, we discuss our main findings for research questions, grounded design recommendation for pairing orchestration tools, and future research direction for dynamic LGF.

5.1 Main Findings for Research Questions

Regarding the feasibility of the pairing policies (RQ1) and how the feasibility may depend on parameters of the pairing policies (RQ2), we found that *averaged across time and sessions,* it is generally feasible (93.6%) to team up struggling students with non-struggling students, the minute they struggle (Policy 1). This result remains true even when a high percentage of students is deemed ineligible for being teamed up with struggling students (Policy 2). Specifically, the average feasibility remains above 80% of struggling students across all sessions unless the pairing restriction rate is above 40%. However, as we see in our case study, there can be sessions and moments with high struggle ratios (hence, low feasibility) when using Policy 1. Relatedly, sessions with very high struggle seem more susceptible to the influence of the PR rate in Policy 2 than a typical session.

When pairing students based on whether their knowledge levels are different (Policy 3) or similar (Policy 4), the policy feasibility is highly dependent on the required KD. For Policy 3, there is a tradeoff between the desired heterogeneity (i.e., the knowledge distance threshold) and the policy's feasibility. This means instructors cannot set a high threshold for the KD if they want to pair most students. In Policy 4, the corresponding tradeoff (between homogeneity in knowledge and policy feasibility) is less prominent. Instructors may choose a stricter ceiling for students' similarity in knowledge levels and still achieve high policy feasibility. In the case study, we found that the policy feasibility in different sessions can be differently influenced by the required KD threshold or ceiling, depending on how closely clustered together students' IK is. For a given, fixed KD threshold (ceiling), a class of students closely clustered IK may result in higher feasibility for Policy 4 and lower feasibility for Policy 3. Presumably, the feasibility of these policies also depends on class size. For example, from our analysis, we hypothesize that for larger classes, the feasibility of pairing policies may change less drastically, when the policy parameters change or as the class progresses.

Regarding policy feasibility by class (RQ3), our results show no significant difference among classes for Policy 1 (Struggle with non-struggle) or Policy 4 (Knowledge similarity pairing). However, we observed significant differences among classes for Policy 2 (Pairing with restriction) and Policy 3 (Knowledge difference pairing). Although different classes have significantly different initial knowledge and struggle status, these differences in IK and struggle status are not always correlated with the feasibility of policies for that class. For example, classes that have different IK may not always have different feasibility for Policy 3 or 4.

5.2 Recommendations for Tool Design

The current study aims to inform the design of an orchestration tool that can help pair students dynamically. We aim to lessen teachers' orchestration load when managing *fluid social transitions*. Such a tool plays a key role in our vision for the smart classroom of the future, in which students alternate fluidly between individual and collaborative learning. Here, we highlight three design implications grounded in findings from the current work. These design implications may inform tools that aim to help teachers manage fluid social transitions, and ensure the feasibility of dynamic LGF policies. It may also offer inspirations, more broadly, for orchestration tools that aim to team up students in CSCL.

Firstly, technology could be used to automatically adjust the parameters used in LGF policies. Our study suggests that the four pairing policies studied provide a promising foundation for an orchestration tool, but greater flexibility is needed to deal with a wide range of circumstances than each individual policy provides. While some policies (e.g., Policies 1 and 2) explored in this study, have a good chance of working well during many class sessions, any given instantiation of a policy (with fixed parameter settings) does not fully deal with class variability and extreme cases. One way to compensate might be to have the tool automatically loosen policy parameters as needed. For example, the tool may gradually loosen the KD threshold or ceiling for policies 3 and 4, when it senses the pairing feasibility to be low.

Secondly, technology could use multiple LGF criteria in cascading fashion, to achieve high feasibility. Specifically, the tool may start out using the ideal pairing policies, and then iteratively try "more loose" criteria if the previous one fails to pair up all students. For example, the tool may first attempt to team up students based on struggle on specific KCs - a criterion that is more specific (and restrictive) than Policy 1, but one that could potentially be more effective for helping struggling students. If that fails, then it might pair up students based on their general struggle (Policy 1). If that fails again then the tool could try to

pair students based on knowledge distance. The tool could also customize its pairing criteria such as using students characteristics that make the most sense in the given classroom context.

Lastly, technology could be used to recommend LGF policies or policy parameters with high feasibility to teachers. Instead of relying solely on the teachers to make pairing decisions, the tool may adopt *SimPairing* to automatically calculate and maximize policies' feasibility based on classroom contexts and recommend them to teachers. For example, if the tool determined, using historical transaction data, that students in a given class have consistently low struggling ratios and fewer wheel-spinning students than non-wheel-spinning ones, it may advise that teachers adopt pairing Policy 1 as it has high feasibility. In addition, our findings open up the potential for the tool to help teachers make informed decisions about parameter configuration, by notifying them of expected feasibility. For example, if a teacher severely restricts the acceptable pairings, the tool could alert teachers of the low feasibility of the pairing policy, and ask if the teacher might want to loosen the restrictions. Running such simulations and providing notification can inform the teachers about the outcomes of the policy feasibility in their classroom, prior to implementing them. This may prevent teachers from choosing policy configurations that are misaligned with their goals.

5.3 Future Research Directions of Dynamic LGF

Building on our investigation, we outline four potential directions for how future research could further explore dynamic LGF policies.

Firstly, in addition to inter-session pairing based on knowledge distance (Policies 3 and 4), future work could explore *intra-session grouping based on knowledge level,* which allows forming pairings during the learning process. Researchers should explore how it would differ from our inter-session grouping and which approach better supports teachers' needs.

Secondly, the current policies identify the students to be wheel-spinning if they struggle on *any* of the KCs. Future work could explore whether teachers prefer to pair students based on their KC-specific struggle status. For example, to help a student struggling on the KC *combine constant terms* in equation solving, teachers may prefer to find a partner who has already mastered the same KC, or at minimum is not struggling on the same KC; they may (or may not) might find it acceptable, if the partner is struggling on another KC, e.g., *divide by variable coefficient*. Relatedly, teaming up students who are both struggling, but struggling on different knowledge components, may have benefits. Such a pair of students may have complementary knowledge and strength, and may help each other get unstuck and stop wheel-spinning. Such pairing criterion opens up good opportunities for tutor-tutee role-switching and mutual peer tutoring.

Thirdly, analogous to pairing based on KC-specific struggle status, instead of using the mean of students' mastery on different KCs to represent their knowledge, future work could explore to what extent *pairing students based on KC-specific knowledge distance* can be more effective, feasible, or preferable for teachers. KC-specific knowledge pairing might be useful for Policy 3, if teachers want two students who have very different skill levels on one specific KC so that the one with higher mastery on that KC can tutor the one with lower mastery.

Lastly, in addition to knowledge level and struggle status, which this work investigated for dynamic grouping, future work can investigate *other student characteristics* (e.g., history of collaborative episodes, preferences for working individually or collaboratively) or other *sources for knowledge level* (e.g., exams or quizzes score) for dynamic pairing. It may also be especially promising to further study pairing based on dynamic student behaviors that can be detected real-time by ITS from interaction data, to allow fluid social transitions and dynamic pairing.

5.4 Limitations

There is uncertainty in the *SimPairing* process in that we do not have a good way of estimating how long any given collaborative episode will last. Thus, *SimPairing* does not simulate students' being unavailable for pairing while they are working collaboratively, until they finish the collaborative episode. There is some reason to think that the resulting inaccuracy in the feasibility results is not severe, as argued, but we do not have a good way of investigating that issue in depth. Additionally, feasibility of pairing policies, while important, is just one piece of the puzzle. It is important, as well, to understand if students *learn better* with these pairing policies (effectiveness). Future research should validate these pairing policies in classroom studies, testing both their effectiveness and feasibility.

6. CONCLUSION

We study the feasibility of pairing policies in the context of ITS, to inform the design of a tool for orchestrating fluid transitions between individual and collaborative learning. Our findings show that on average, dynamically pairing students based on their in-the-moment wheel-spinning status results in good pairing feasibility for struggling students on average, even with moderate restrictions on the allowed pairings. We also found the trade-off between the *required knowledge distance* and *the policy feasibility,* is more prominent in heterogeneous grouping than in homogeneous grouping. However, any given instantiation of a policy (with fixed parameter settings) does not fully deal with class variability and extreme cases, as policies have different feasibility for different classes and sessions. This suggests optimization for policy feasibility (e.g., through gradually loosening parameters) or classroom customization need to be taken into consideration. Methodologically, this research extends previous work (e.g., Replay Enactments) that used authentic data and algorithms as design materials to augment designers' intuitions for designing future tools [27].

This work has several novel elements. First, using the *SimPairing* approach, our work explores the *feasibility* of LGF policies derived from user research with math teachers. In addition, to the best of our knowledge, this is the first study that considers students' in-the-moment wheel-spinning status in dynamic pairing policies. Finally, our work addresses a gap in the literature for dynamic intra-session LGF [1] and envisions how instructors and/or an orchestration tool will customize pairing policies and parameters to specific classroom contexts, which prior work argued to be especially helpful in the LGF process [1,8].

In sum, theoretically, this work bridges the literature gap on its investigation of the *feasibility* of user-centered dynamic pairing policies. Practically, we contribute grounded design directions for pairing orchestration tools, and *SimPairing* as an approach, to evaluate dynamic LGF policies, which may generalize to other online educational software that have transaction data.

7. ACKNOWLEDGEMENTS

This work was supported in part by Grant #1822861 from the National Science Foundation (NSF). Any opinions presented in this article are those of the authors and do not represent the views of the NSF. We thank Yanjin Long and Dr. Zach Branson for their help, and the anonymous reviewers for their feedback.

8. REFERENCES

[1] S. Amara, J. Macedo, F. Bendella, A. Santos, Group formation in mobile computer supported collaborative learning contexts: A systematic literature review. Journal of Educational Technology & Society. 19 (2016) 258–273.

[2] P. Dillenbourg. Over-scripting CSCL: The risks of blending collaborative learning with instructional design. P. A. Kirschner. Three worlds of CSCL. Can we support CSCL?, Heerlen, Open Universiteit Nederland. 61-91, 2002.

[3] X. Wang, M. Thompson, K. Yang, D. Roy, K.R. Koedinger, C.P. Rose, J. Reich, Practice-based teacher questioning strategy training with ELK: A role-playing simulation for eliciting learner knowledge. Proc. ACM Hum.-Comput. Interact. 5 (2021) 1–27.

[4] Y.-M. Huang, Y.-W. Liao, S.-H. Huang, H.-C. Chen, Jigsaw-based cooperative learning approach to improve learning outcomes for mobile situated learning. Journal of Educational Technology & Society. 17 (2014) 128–140.

[5] N.M. Webb, K.M. Nemer, A.W. Chizhik, B. Sugrue, Equity Issues in collaborative group assessment: group composition and performance. Am. Educ. Res. J. 35 (1998) 607–651.

[6] I. Srba, M. Bielikova, Dynamic group formation as an approach to collaborative learning support. IEEE Trans. Learn. Technol. 8 (2015) 173–186.

[7] A. Mujkanovic, D. Lowe, K. Willey, C. Guetl, Unsupervised learning algorithm for adaptive group formation: Collaborative learning support in remotely accessible laboratories. International Conference on Information Society (i-Society 2012), 50–57.

[8] N. Maqtary, A. Mohsen, K. Bechkoum, Group formation techniques in computer-supported collaborative learning: A systematic literature review. Technology, Knowledge and Learning. 24 (2019) 169–190.

[9] K. Holstein, B.M. McLaren, V. Aleven, Designing for complementarity: teacher and student needs for orchestration support in AI-Enhanced classrooms. Artificial Intelligence in Education. Springer International Publishing, (2019).157–171.

[10] J.K. Olsen, N. Rummel, V. Aleven, Designing for the co-orchestration of social transitions between individual, small-group and whole-class learning in the classroom. International Journal of Artificial Intelligence in Education. (2020) 24-56

[11] N. Ding, R.J. Bosker, E.G. Harskamp, Exploring gender and gender pairing in the knowledge elaboration processes of students using computer-supported collaborative learning. Comput. Educ. 56 (2011) 325–336.

[12] M.E. Lockheed, A.M. Harris, Cross-sex collaborative learning in elementary classrooms. American Educational Research Journal. 21 (1984) 275–294.

[13] D.A. Sandmire, P.F. Boyce, Pairing of opposite learning styles among allied health students: effects on collaborative performance, J. Allied Health. 33 (2004) 156–163.

[14] Y.-C. Kuo, H.-C. Chu, C.-H. Huang, A learning style-based grouping collaborative learning approach to improve EFL students' performance in English courses. Journal of Educational Technology & Society. 18 (2015) 284–298.

[15] E. Alfonseca, R.M. Carro, E. Martín, A. Ortigosa, P. Paredes, The impact of learning styles on student grouping for collaborative learning: a case study. User Model. User-Adapt Interact. 16 (2006) 377–401.

[16] Y. Taniguchi, Y. Gao, K. Kojima, S. Konomi, Evaluating learning style-based grouping strategies in real-world Collaborative Learning Environment, Distributed, Ambient and Pervasive Interactions: Technologies and Contexts. (2018) 227–239.

[17] P.-J. Chuang, M.-C. Chiang, C.-S. Yang, C.-W. Tsai, Social networks-based adaptive pairing strategy for cooperative learning. Journal of Educational Technology & Society. 15 (2012) 226–239.

[18] E.A. Day, W. Arthur, S.T. Bell, B.D. Edwards, W. Bennett, J.L. Mendoza, T.C. Tubré, Ability-based pairing strategies in the team-based training of a complex skill: Does the intelligence of your training partner matter? Intelligence. 33 (2005) 39–65.

[19] R. Niu, L. Jiang, Y. Deng, Effect of Proficiency Pairing on L2 Learners' Language Learning and Scaffolding in Collaborative Writing. The Asia-Pacific Education Researcher. 27 (2018) 187–195.

[20] J.A. Sutherland, K.J. Topping, Collaborative creative writing in eight−year−olds: Comparing cross−ability fixed role and same−ability reciprocal role pairing. J. Res. Read. 22 (1999) 154–179.

[21] N. Storch, A. Aldosari, Pairing learners in pair work activity, Language Teaching Research. 17 (2013) 31–48.

[22] L.S. Fuchs, D. Fuchs, C.L. Hamlett, K. Karns, High-achieving students' interactions and performance on complex mathematical tasks as a function of homogeneous and heterogeneous pairings. American Educational Research Journal. 35 (1998) 227–267.

[23] H.-W. Tien, Y.-S. Lin, Y.-C. Chang, C.-P. Chu, A genetic algorithm-based multiple characteristics grouping strategy for collaborative learning. International Conference on Web-Based Learning, Springer, (2013) 11–22.

[24] Y. Pang, F. Xiao, H. Wang, X. Xue, A Clustering-Based Grouping Model for Enhancing Collaborative Learning. 13th International Conference on Machine Learning and Applications. (2014) 562–567.

[25] B. Chen, G. Hwang, T. Lin, Impacts of a dynamic grouping strategy on students' learning effectiveness and experience value in an item bank−based collaborative practice system. Br. J. Educ. Technol. 51 (2020) 36–52.

[26] Y.-T. Lin, Y.-M. Huang, S.-C. Cheng, An automatic group composition system for composing collaborative learning groups using enhanced particle swarm optimization, Comput. Educ. 55 (2010) 1483–1493.

[27] K. Holstein, E. Harpstead, R. Gulotta, J. Forlizzi, Replay Enactments: Exploring possible futures through historical data. Proceedings of the 2020 ACM Designing Interactive Systems Conference, Association for Computing Machinery, New York, NY, USA, (2020)1607–1618.

[28] T. Nagashima, K. Yang, A. Bartel, E. Silla, N. Vest, M. Alibali, V. Aleven, Pedagogical Affordance Analysis: Leveraging teachers' pedagogical knowledge to elicit pedagogical affordances and constraints of instructional tools. International Society of the Learning Sciences (2020)

[29] N. Diana, M. Eagle, J. Stamper, Automatic peer tutor matching: Data-driven methods to enable new opportunities for help, (n.d.).

[30] V. Echeverria, K. Holstein, J. Huang, J. Sewall, N. Rummel, V. Aleven, Exploring human–AI control over dynamic transitions between individual and collaborative learning. In European Conference on Technology Enhanced Learning, Springer, Cham. (2020) 230–243.
[31] K.B. Yang, L. Lawrence, V. Echeverria, B. Guo, K. Holstein, N. Rummel, V. Aleven. (Under Review). "I like student choice, program insights, but final say from the teacher": Teachers' Preferences regarding Human-AI Control in Dynamic Student Pairing. Manuscript submitted to EC-TEL 2021
[32] J.E. Beck, Y. Gong, Wheel-spinning: students who fail to master a skill, in: Artificial Intelligence in Education, Springer Berlin Heidelberg, 2013: 431–440.
[33] S. Kai, M.V. Almeda, R.S. Baker, C. Heffernan, N. Heffernan, Decision tree modeling of wheel-spinning and productive persistence in skill builders. Journal of Educational Data Mining. 10 (2018) 36–71.
[34] N. Matsuda, S. Chandrasekaran, J.C. Stamper, How quickly can wheel spinning be detected? Educational Data Mining, ERIC (2016) 607–608.
[35] C. Zhang, Y. Huang, J. Wang, D. Lu, W. Fang, J. Stamper, S. Fancsali, K. Holstein, V. Aleven, Early detection of wheel spinning: comparison across tutors, models, features, and operationalizations, International Educational Data Mining Society. (2019).
[36] T. Mu, A. Jetten, E. Brunskill, Towards suggesting actionable interventions for wheel-spinning Students. The 13th International Conference on Educational Data Mining, 183–193.
[37] K. Holstein, B.M. McLaren, V. Aleven, Co-designing a real-time classroom orchestration tool to support teacher–AI complementarity. Journal of Learning Analytics. 6 (2019) 27–52.
[38] N. Diana, M. Eagle, J. Stamper, S. Grover, M. Bienkowski, S. Basu, Peer tutor matching for introductory programming: Data-driven methods to enable new opportunities for help. International Society of the Learning Sciences. (2018)
[39] K.B. Yang, T. Nagashima, J. Yao, J.J. Williams, K. Holstein, V. Aleven. Can Crowds Customize Instructional Materials with Minimal Expert Guidance? Exploring Teacher-guided Crowdsourcing for Improving Hints in an AI-based Tutor. Proc. ACM Hum.-Comput. Interact. 5 (2021) 1–24.
[40] S. Ritter, J.R. Anderson, K.R. Koedinger, A. Corbett, Cognitive tutor: applied research in mathematics education, Psychon. Bull. Rev. 14 (2007) 249–255.
[41] Y. Long, V. Aleven, Supporting students' self-regulated learning with an open learner model in a linear equation tutor, in: International Conference on Artificial Intelligence in Education, Springer, (2013) 219–228.
[42] K. Holstein, B.M. McLaren, V. Aleven, Student Learning Benefits of a Mixed-Reality Teacher Awareness Tool in AI-Enhanced Classrooms. Artificial Intelligence in Education, Springer International Publishing, (2018) 154–168.
[43] M. Waalkens, V. Aleven, N. Taatgen, Does supporting multiple student strategies lead to greater learning and motivation? Investigating a source of complexity in the architecture of intelligent tutoring systems, Computers & Education. 60 (2013) 159–171.
[44] A.T. Corbett, J.R. Anderson, Knowledge tracing: Modeling the acquisition of procedural knowledge, User Model. User-Adapt Interact. 4 (1995) 253–278.
[45] E. Walker, N. Rummel, K.R. Koedinger, Adaptive intelligent support to improve peer tutoring in algebra. International Journal of Artificial Intelligence in Education. 24 (2014) 33–61.
[46] E. Walker, N. Rummel, K.R. Koedinger, To Tutor the Tutor: Adaptive Domain Support for Peer Tutoring, Intelligent Tutoring Systems. 626–635.
[47] K.R. Koedinger, J. Stamper, P.F. Carvalho, Sharing and Reusing Data and Analytic Methods with LearnSphere, Hands-On, 2, 30p.
[48] M.V.Q. Almeda, When practice does not make perfect: Differentiating between productive and unproductive persistence. (2018). Doctoral dissertation, Columbia University.
[49] Y.-M. Huang, T.-T. Wu, A systematic approach for learner group composition utilizing U-learning portfolio, Educational Technology & Society, Vol. 14, (2011).
[50] D.W. Johnson, R.T. Johnson, Learning together and alone: Cooperative, competitive, and individualistic learning, 2nd ed, 2 (1987) 193.
[51] N.M. Webb, A.S. Palincsar, Group processes in the classroom, in: D.C. Berliner (Ed.), Handbook of Educational Psychology , (pp, Macmillan Library Reference Usa; London, England, New York, NY, US, (1996). 841–873.
[52] R. Hübscher, Assigning Students to Groups Using General and Context-Specific Criteria, IEEE Trans. Learn. Technol. 3 (2010) 178–189.

Learning from Non-Assessed Resources: Deep Multi-Type Knowledge Tracing

Chunpai Wang, Siqian Zhao, Shaghayegh Sahebi
Department of Computer Science
University at Albany-SUNY
Albany, NY 12222
{cwang25,szhao2,ssahebi}@albany.edu

ABSTRACT

The state of the art knowledge tracing approaches mostly model student knowledge using their performance in assessed learning resource types, such as quizzes, assignments, and exercises, and ignore the non-assessed learning resources. However, many student activities are non-assessed, such as watching video lectures, participating in a discussion forum, and reading a section of a textbook, all of which potentially contributing to the students' knowledge growth. In this paper, we propose the first novel deep learning based knowledge tracing model (DMKT) that explicitly model student's knowledge transitions over both assessed and non-assessed learning activities. With DMKT we can discover the underlying latent concepts of each non-assessed and assessed learning material and better predict the student performance in future assessed learning resources. We compare our propose method with various state of the art knowledge tracing methods on four real-world datasets and show its effectiveness in predicting student performance, representing student knowledge, and discovering the underlying domain model.

Keywords

Knowledge Tracing, Multiple Learning Resource Types, Non-Assessed Learning Resources, Memory Augmented Neural Networks, Domain Knowledge Modeling, Student Knowledge Modeling

1. INTRODUCTION

As the education landscape shifts toward distance learning, the online learning systems advance in complexity and capacity. They can handle more students, evaluate students through different kinds of assessments, and offer various types of learning resources to them. In such systems, a student can study a reading section, take a quiz, watch a video lecture, and practice programming in an embedded development environment. As a result, students learn from heterogeneous types of activities in modern online learning systems, among which some can be assessed and some cannot.

Despite this heterogeneity in learning resource types, current student knowledge tracing models mostly focus on assessed learning resources, ignoring the non-assessed ones. In the assessed learning resource types, such as quizzes and assignments, students' performance can be evaluated given their answers and solutions. These kinds of learning resources provide a window to student knowledge through observing their performance. Conversely, in the non-assessed learning resources, such as readings and video lectures, such an observation does not exist. Hence, evaluating student knowledge and performance while interacting with these learning resources is a difficult task [4, 11, 10].

Indeed, because current knowledge tracing approaches do not model non-assessed learning resources, identifying their underlying concepts, finding the similarities between these learning resources, and in general domain knowledge modeling for such non-assessed learning materials is still a challenging problem. That is, many modern knowledge tracing models do not rely on a predefined domain knowledge model, such as a Q-matrix, and can identify the "latent concepts" that are being evaluated in problems, quizzes, or assignments [20, 23, 21, 7, 9]. This is particularly useful when annotating learning materials with their concepts is expensive or infeasible. However, discovering such latent concepts in non-assessed learning resources is an under-explored research area. Some recent works have aimed in identifying such latent concepts [19] and similarities [17] between assessed and non-assessed learning materials. However, their findings were according to static student performance, ignoring the sequential learning data of students.

In this paper, we argue that modeling non-assessed learning materials is essential and non-dispensable in tracing student knowledge. Students learn from all types of activities and ignoring a large portion of student activities is a missed opportunity in student knowledge tracing. Especially that previous research has shown that working with various learning activity types has considerable benefits for student learning [15, 2, 1, 12]. Hence, modeling both assessed and non-assessed learning activities should result in a more accurate estimation of student knowledge state and prediction of their performance on future assessed learning resources.

Accordingly, we propose Deep Multi-type Knowledge Trac-

Chunpai Wang, Siqian Zhao and Shaghayegh Sahebi "Learning from Non-Assessed Resources: Deep Multi-Type Knowledge Tracing". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 195-205. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

ing (DMKT) model, which not only traces student knowledge states over various learning activity types but also provides a feasible solution to discovering underlying patterns or concepts for both assessed and non-assessed learning resources. To this end, DMKT estimates student knowledge gain between every two consecutive assessed learning activities according to student performance on them. At the same time, it distributes this estimated knowledge gain among the in-between non-assessed learning activities and the latest assessed activity. We use an attention mechanism for this distribution. As a result, DMKT can model the underlying latent concepts for each of the assessed and non-assessed learning resources, evaluate student knowledge after interacting with these learning resources and predict student performance on the assessed ones.

We evaluate our proposed model on four real-world datasets, showing the significant effect of modeling various learning resource types on the task of student performance prediction. Also, we showcase the interpretability of DMKT by visualizing student knowledge while working with various learning resource types. Finally, we demonstrate the power of DMKT in discovering the learning resources' similarities and underlying latent concepts.

2. RELATED WORK

Student knowledge tracing aims to capture the student's knowledge state and knowledge state transition patterns, which could be further used for tasks like students' performance prediction, intelligent curriculum design, and interpretation and discovery of structure in student tasks.

Traditional knowledge tracing methods modeled knowledge transition on assessed learning resources using predefined domain knowledge models (concepts of learning resources). For example, Drasgow et al. proposed IRT that leverages the structured logistic regression to model student's dichotomous responses and estimates the student's ability, learning resource difficulty [8]. BKT uses binary variables for modeling whether student acquires a concept or not, and a Hidden Markov Model is used to update the probability that student answers a question correctly [6, 22]. However, since annotating a domain knowledge model can be expensive and time consuming, in many real-world scenarios, such predefined domain knowledge models are not be provided. To solve this problem, new approaches turn to investigate modeling student knowledge and domain knowledge at the same time. For example, Lan et al. utilized the matrix factorization to model the student knowledge and concept-question association, assuming the sparse association between concepts and questions [14]. As another example, Doan et al. model student learning with a tensor factorization in which the student knowledge is having an increasing trend using a rank-based constraint [7].

At the same time, in the past few years with the advance of deep neural networks, deep knowledge tracing methods have emerged. For example, DKT [18] utilizes LSTM to model students' knowledge transition over time. Recently, transformer-based neural networks have been successfully applied to model the different knowledge transitions of different students' historical interactions on learning resources [5, 9]. SAKT [16] uses the self-attention mechanism to model the interdependencies among interactions on the sequence. In [23], Zhang et al. proposed a Dynamic Key-Value Memory Networks based method (DKVMN), which integrates the memory augmented neural networks with the attention mechanism, to exploit the relationships between underlying concepts for better students' skill acquisition modeling. Yeung et al. extended DKVMN, by integrating the one-parameter logistic item response theory to provide better interpretability [21]. However, none of the deep knowledge tracing models have focused on modeling the non-assessed learning activities and tracing student knowledge on such activities.

Knowledge Tracing using Multiple Learning Resource Types. Previous approaches ignored the effect of learning activities on non-assessed learning resources, none of the methods mentioned above consider both assessed and non-assessed learning resources at the same time. However, in reality, students not only learn from practicing assessed learning resources (such as questions) but also learn by studying the non-assessed one, such as watching video lectures, reading textbooks, and discussing with others. One reason for not modeling the non-assessed activities is that reliable student performance observations are missing in these activities. This makes modeling the knowledge transition from these non-assessed learning activities difficult. To the best of our knowledge, the only existing work that models non-assessed learning activities along with the assessed ones is Multi-View Knowledge Model (MVKM) [25]. MVKM models multiple learning resources jointly using tensor factorization to capture latent students' features and latent learning resource concepts, assuming that latent concepts are shared by different learning resource types. However, this method can only capture the linear dependencies between variables, as the latent students' features and latent learning resource concepts are multiplied via linear matrix and tensor products. On the other hand, due to the large memory cost of tensor factorization, MVKM can not handle the datasets with very large student and learning resource numbers. Unlike MVKM, our proposed method in this paper considers the non-linear relationships between variables, and handles large datasets, while modeling student knowledge gain from multiple learning resource types (both assessed and non-assessed).

3. DEEP MULTI-TYPE KNOWLEDGE TRACING (DMKT)

3.1 Problem Formulation

A standard knowledge tracing (KT) problem is to predict student performance or response on an upcoming question, given the learner's performance records on previously solved questions. These records typically consist of a sequence of questions and responses at each discrete time step, denoted as a tuple (q_t^s, r_t^s) for student s at time step t. Since we only discuss how to predict future performance for a single student, we omit the superscript s in the following sections. Therefore, given students' past history records up to time $t-1$ as $\{(q_1, r_1), \cdots, (q_{t-1}, r_{t-1})\}$, the goal of KT is to predict their response r_t to question question q_t at the current time step t.

In this paper, we aim to incorporate students' non-assessed

learning activities and model student knowledge transition over both assessed and non-assessed learning resources, such as solving quizzes, watching video lectures, viewing annotated examples or hints, and participating discussion forums. Therefore, given student's past historical responses to assessed learning materials as well as past history of non-assessed learning activities, we would like to estimate student knowledge and predict their performance in the next assessed learning resource. To do this, assuming L distinct non-assessed learning resources and Q distinct assessed ones, we represent students' historical records up to time $t-1$ as $\{(q_1, r_1), \mathcal{L}_1, (q_2, r_2), \mathcal{L}_2, \cdots, (q_{t-1}, r_{t-1}), \mathcal{L}_{t-1}\}$, in which $\mathcal{L}_t = \{l_t^1, l_t^2, \cdots, ...l_t^n\}$ at each time step t denotes the sequence of n non-assessed learning activities (e.g., watching video lectures) between the assessed activities (e.g., answering questions) q_t and q_{t+1}. Our goal is to predict student performance on assessed learning material q_t at each time step t, model student knowledge at and between time steps in interaction with q_t and all l_t^is, and discover the underlying latent concepts of assessed qs and non-assessed l^is.

3.2 The Base Model

We base our DMKT model upon a recent successful deep knowledge tracing model: DKVMN [23]. DKVMN is a special type of memory-augmented neural networks (MANN) for knowledge tracing which has one static key matrix to store the knowledge concepts and one dynamic value matrix to store students' updated mastery levels of those corresponding concepts. Assuming that there are N latent concepts $\{c^1, \cdots, c^N\}$ for each learning resource, and each latent concept can be represented by d_h-dimensional embeddings, similar to DKVMN, DMKT has the key matrix $\mathbf{M}^k$ of size $N \times d_h$ to store the N knowledge concepts. Similarly, the value matrix $\mathbf{M}_t^v$ of size $N \times d_h$ stores the student's mastery levels of each concept, at time step t.

However, DKVMN only supports updating knowledge states $\mathbf{M}_t^v$ on assessed learning materials, and lacks the ability to leverage the abundant of data other than student responses on assessed learning materials. To overcome this limitation, our proposed DMKT updates $\mathbf{M}_t^v$ with an additional internal component that employs the attention mechanism to process the non-assessed learning activities between any two assessed ones and use the updated $\mathbf{M}_t^v$ to predict student's performance on upcoming assessed learning resource. This component contains two functionalities, one is to update student knowledge state on non-assessed learning activities, and another is to summarize all activity contexts before an assessed activity to help accurate prediction of student performance.

One may think that a straightforward solution to integrate the non-assessed learning resources would be to consider them as student interaction features. However, since the non-assessed learning activities are not explicitly represented in such models, their contribution to student knowledge could be assessed. Also, such an approach cannot model student's knowledge transition between different non-assessed learning activities. In the following, we introduce our novel updating and summarizing functionalities that help DMKT to model all learning activity types. An overview of DMKT's architecture can be found in Figure 1[1].

3.3 Learning Resource Attention Weights

For the simplicity of illustration, let us assume that there is only one non-assessed learning activity, e.g., watching a video lecture, between solving two problems q_{t-1} and q_t, that is $\mathcal{L}_{t-1} = \{l_{t-1}\}$. DMKT assumes that student knowledge gets updated as the student interacts with l_{t-1} and q_t, weighted by their corresponding attention weights. So, in each step, DMKT uses attention weights from q_t and l_{t-1} to update the student knowledge in the concepts' embeddings, $\mathbf{M}_t^v$.

To compute the attention weights, DMKT first embeds all questions into an embedding matrix $\mathbf{A}^q \in \mathbb{R}^{Q \times d_h}$, and all video lectures in another embedding matrix $\mathbf{A}^l \in \mathbb{R}^{L \times d_h}$. At each time step, DMKT extracts the embedding vector of q_t ($\mathbf{k}_t \in \mathbb{R}^{d_h}$) from $\mathbf{A}^q$, as well as the embedding vector $\mathbf{k}_{t-1}^l \in \mathbb{R}^{d_h}$ of l_{t-1} from $\mathbf{A}^l$. Then, it uses these embedding vectors to query the key memory matrix $\mathbf{M}^k$ to obtain the attention weights $w_t^q(i)$ and $w_t^l(i)$ respectively as follows:

$$w_t^q(i) = \text{Softmax}\left(\mathbf{k}_t^{q\top} \mathbf{M}^k(i)\right) \tag{1}$$

$$w_{t-1}^l(i) = \text{Softmax}\left({\mathbf{k}_{t-1}^l}^{\top} \mathbf{M}^k(i)\right) \tag{2}$$

The attention weight in $\mathbf{w}_t^q$ and $\mathbf{w}_{t-1}^l$ can be viewed as respectively the correlation between question q_t and lecture l_{t-1} with each of the N latent concepts. Notice that, $w_t^q(i)$ and $w_{t-1}^l(i)$ are the i-th element in the attention weight vectors $\mathbf{w}_t^q$ and $\mathbf{w}_{t-1}^l$ respectively, and for interpretability purposes the attention weights sum to one ($\sum_{i=1}^N w_t^q(i) = \sum_{i=1}^N w_{t-1}^l(i) = 1$).

3.4 Student Performance Prediction

At each time step t, DMKT aims to predict the student's performance on q_t. Since the predicted performance is a result of student knowledge that is gained by interacting with both problems and lectures, it is intuitive to aggregate these knowledge gains and predict the student performance accordingly. Remember that the memory value matrix $\mathbf{M}_t^v \in \mathbf{R}^{N \times d_h}$ is used to represent student's knowledge state on each concept embedding. So, to summarize the student's mastery level of question q_t and lecture l_{t-1} in the N concepts, we compute the weighted sum of all memory slots in the value matrix using attention weight vectors $\mathbf{w}_t^q$ and $\mathbf{w}_{t-1}^l$, respectively.

$$\mathbf{r}_t^q = \sum_{i=1}^{N} w_t^q(i) \mathbf{M}_t^v(i) \tag{3}$$

$$\mathbf{r}_{t-1}^l = \sum_{i=1}^{N} w_{t-1}^l(i) \mathbf{M}_t^v(i) \tag{4}$$

Then, we concatenate the latent knowledge states or mastery levels $\mathbf{r}_t^q$ and $\mathbf{r}_{t-1}^l$ on question q_t and lecture l_{t-1} with question embedding $\mathbf{k}_t^q$ as well as lecture embedding $\mathbf{k}_{t-1}^l$

[1]The source code is provided at: `https://github.com/persai-lab/EDM2021-DMKT`

vertically and pass them into a fully connected layer with a Tanh activation to obtain a summary vector $\mathbf{f}_t$

$$\mathbf{f}_t = \text{Tanh}\left(\mathbf{W}_1^\top \left[\mathbf{r}_t^q, \mathbf{r}_{t-1}^l, \mathbf{k}_t^q, \mathbf{k}_{t-1}^l\right] + \mathbf{b}_1\right) \tag{5}$$

where $[\cdot]$ denotes concatenation. This summary vector $\mathbf{f}_t$ contains a summary of all information, such as student ability and the relationship between question q_t and lecture l_{t-1}, to predict student response at time t accurately. Finally, the student's performance in query question q_t is calculated by passing the feature vector $\mathbf{f}_t$ through another fully connected layer with a Sigmoid activation as follows:

$$p_t = \text{Sigmoid}\left(\mathbf{W}_2^\top \mathbf{f}_t + \mathbf{b}_2\right) \tag{6}$$

3.5 Student Knowledge Update

DMKT tracks the student knowledge states by updating the memory value matrix $\mathbf{M}_t^v$ after each learning activity on q_t and l_t so as to predict student performance on q_{t+1} using the updated $\mathbf{M}_{t+1}^v$.

For assessed learning activities, we first retrieve an embedding vector of (q_t, r_t), denoted by $\mathbf{v}_t^q \in \mathbb{R}^{d_h}$, from a response embedding matrix $\mathbf{B}$ of size $2Q \times d_h$. This embedding $\mathbf{v}_t$ contains the information about how much student knowledge should be updated after working on question q_t with outcome r_t^q. We also use the *erase-followed-by-add* mechanism to update the memory value matrix, that is to erase the memory first using erase vector $\mathbf{e}_t^q \in [0,1]^{d_h}$ before adding new information with the add vector $\mathbf{a}_t^q \in \mathbb{R}^{d_h}$. This update of each value memory slot could be summarized as an erase step and an add step as follows:

Erase Step:

$$\begin{aligned} \mathbf{e}_t^q &= \text{Sigmoid}\left(\mathbf{E}^\top \mathbf{v}_t^q + \mathbf{b}_\mathbf{e}^q\right) \\ \tilde{\mathbf{M}}_t^v(i) &= \mathbf{M}_{t-1}^v(i) \otimes [\mathbf{1} - w_t^q(i)\mathbf{e}_t^q] \end{aligned} \tag{7}$$

Add Step:

$$\begin{aligned} \mathbf{a}_t^q &= \text{Tanh}\left(\mathbf{D}^\top \mathbf{v}_t^q + \mathbf{b}_\mathbf{a}^q\right)^T \\ \mathbf{M}_t^v(i) &= \tilde{\mathbf{M}}_{t-1}^v(i) + w_t^q(i)\mathbf{a}_t^q \end{aligned} \tag{8}$$

where $\mathbf{1}$ is a vector of all ones, and $\otimes$ represents the element-wise multiplication.

For each non-assessed activity, we follow a similar erase-followed-by-add steps in Eq.(7) and Eq.(8), except that we use $\mathbf{k}_t^l$ directly instead of a new response embedding.

Erase Step on Non-assessed Resources:

$$\begin{aligned} \mathbf{e}_t^l &= \text{Sigmoid}\left(\mathbf{H}^\top \mathbf{k}_t^l + \mathbf{b}_\mathbf{e}^l\right) \\ \tilde{\mathbf{M}}_t^v(i) &= \mathbf{M}_{t-1}^v(i) \otimes \left[\mathbf{1} - w_t^l(i)\mathbf{e}_t^l\right] \end{aligned} \tag{9}$$

Add Step on Non-assessed Resources:

$$\begin{aligned} \mathbf{a}_t^l &= \text{Tanh}\left(\mathbf{G}^\top \mathbf{k}_t^l + \mathbf{b}_\mathbf{a}^l\right)^T \\ \mathbf{M}_t^v(i) &= \tilde{\mathbf{M}}_{t-1}^v(i) + w_t^l(i)\mathbf{a}_t^l \end{aligned} \tag{10}$$

3.6 Network Architecture and Extension

The neural network architecture of DMKT is shown in Figure 1. For illustration simplicity, this figure assumes that there is only one non-assessed learning resource l_t between q_t and q_{t+1}. This architecture mainly contains two components: *read* component for making a prediction on input question q_t and *write* component for updating the value matrix after interacting with l_t and q_t.

When there are multiple non-assessed learning activities between q_t and q_{t+1}, that is $\mathcal{L}_t = \{l_t^1, \cdots, l_t^n\}$, we can simply extend the model by looping over each activity to generate $\mathbf{k}_t^{l^i}$ as well as $\mathbf{r}_t^{l^i}$ using equation (4) for $i \in \{1, \cdots, n\}$. When making predictions, we use $\sum_{i=1}^n \mathbf{k}_t^{l^i}$ to represent $\mathbf{k}_t^l$ and $\sum_{i=1}^n \mathbf{r}_t^{l^i}$ to represent $\mathbf{r}_t^l$ in the architecture. When updating the knowledge, the value matrix is updated sequentially over all activities as described in the previous subsection.

3.7 Training

All learnable parameters , i.e. $\mathbf{A}^q, \mathbf{A}^l, \mathbf{B}$, in the entire DMKT model are trained in end-to-end manner by minimizing the binary cross-entropy loss of all students' assessed responses, i.e.,

$$\ell_{BCE} = -\sum_t \left(o_t \log p_t + (1 - o_t) \log (1 - p_t)\right) \tag{11}$$

where o_t denotes the observation of correctness on assessed response at time t and p_t denotes the prediction of correctness of DMKT at time t.

3.8 Knowledge State Calculation

DMKT is capable of tracing and depicting knowledge concept mastery level for each student. A student's knowledge state before each assessed or non-assessed learning activity can be obtained in the *read* process using the following steps.

Assume that there are N dummy query questions q^is, each of them only using one concept, for the purpose of knowledge state calculation. Each of dummy questions can obtain a designed embedding $\mathbf{k}^i$ such that the correlation weight vector $\mathbf{w}^i$ is "one-hotted", that is $\mathbf{w}^i = [0, \cdots, w_i, \cdots, 0]$ where w_i of concept c^i is equal to 1. Then, we can use each of these one-hot correlation weight vectors to access value matrix state on each slot $\mathbf{M}_t^v(i)$ to obtain $\mathbf{r}_t^i$ for each concept c^i. In other words, $\mathbf{r}_t^i = \mathbf{M}_t^v(i)$ for q^i.

Then, we can predict the student knowledge purely based on $\mathbf{r}_t^i$ by masking the weight of the input content embedding in Eq. (5), which ends up as:

$$\mathbf{f}_t^i = \text{Tanh}\left(\left[\mathbf{W}_1^{r^i}, \mathbf{0}, \mathbf{0}, \mathbf{0}\right]^\top \left[\mathbf{r}_t^i, \mathbf{r}_{t-1}^l, \mathbf{k}_t^i, \mathbf{k}_{t-1}^l\right] + \mathbf{b}_1\right) \tag{12}$$

where $\mathbf{W}_1$ is split into four parts including $\mathbf{W}_1^{r^i}$, $\mathbf{W}_1^{k^i} = \mathbf{0}$, $\mathbf{W}_1^{r^l} = \mathbf{0}$, and $\mathbf{W}_1^{k^l} = \mathbf{0}$. Finally, a scalar value p^i is output as in Eq.(6) to be the predictive mastery level of concept c^i. We repeat this process N times with N numbers of one-hot correlation weight vectors to obtain student's knowledge state vector with size $1 \times N$ after each learning activity.

4. EXPERIMENTS

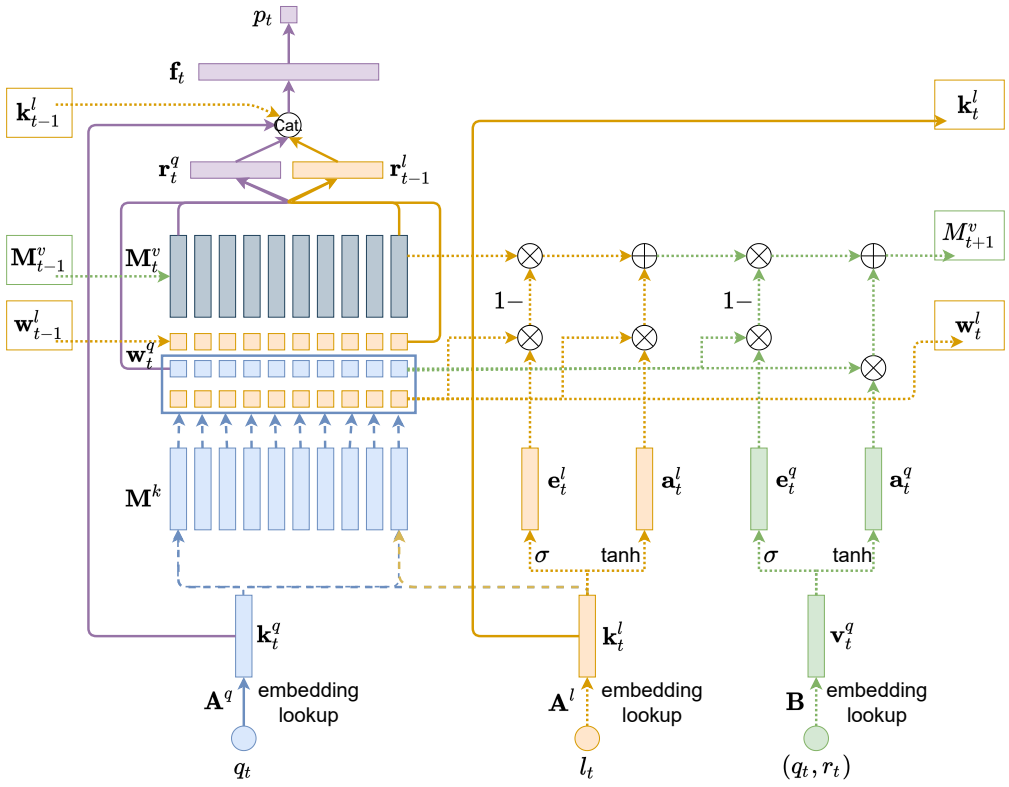


Figure 1: Neural Network Architecture of DMKT.

To evaluate our proposed model, we conduct three kinds of experiments. First, we compare it with state-of-the-art baselines in the student performance prediction task. Second, we analyze the discovered student knowledge transition patterns in terms of assessed and non-assessed learning activities. Last but not least, we validate the non-assessed learning resources' latent concepts discovered by the proposed method.

4.1 Datasets

We use three real-world datasets to evaluate the proposed model:

MORF[2] is an open online course dataset from Coursera [3]. In this course, students can watch lecture videos and work on problems. Each problem is a full complex course assignment. These video lectures and assignments are published in sequential order in this dataset, but students can have multiple attempts on each assignment and watch any video at any time. Students' scores are normalized into $[0, 1]$.

EdNet[3] is collected by Santa[4], a multi-platform AI tutoring service for students to prepare TOEIC English testing. We use the problem explanation documents as the non-assessed learning resources. There are $297,915$ user records in the full dataset, and we randomly extract $1,000$ users' records for experiments.

Junyi[5] is a dataset that comes from a Chinese e-learning website. Students work on problems from 8 math areas. Each problem has several hints, students can request hints when solving problems. We consider the problems as the assessed learning resources and the associated problem hints as the non-assessed learning resources. There are $25,925,922$ records in total from $247,606$ users in the full dataset. We extract two subsets of this full dataset for experiments. One is called Junyi2063, which contains 2063 users' records on 3760 questions and 1432 hints. A smaller dataset named Junyi1564, which consists of 1564 users' records on 142 questions and 116 hints, is extracted to serve the purpose of visualization on concept discovery results. The descriptive statistics of these four datasets are shown in the table 1.

4.2 Baseline Methods

In experiments of performance prediction, we compare with 13 baseline methods on the task of student performance prediction on assessed learning resources, including six state-of-the-art deep learning based knowledge tracing models, one existing tensor factorization based knowledge tracing model supporting multiple learning resource types, and seven extended deep learning based models utilizing non-assessed learning resources as additional input features. These methods are:

[2] https://educational-technology-collective.github.io/morf/
[3] https://github.com/riiid/ednet
[4] https://aitutorsanta.com/intro
[5] https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=1275

Table 1: Descriptive Statistics of 3 Datasets.

Dataset	Users	Questions	Question Records	Mean Question Responses	STD Question Responses	Correct Question Responses	Incorrect Question Responses	Non-gradable Materials	Non-gradable Records
MORF	686	10	12031	0.7763	0.2507	N/A	N/A	52	41980
EdNet	1000	11249	200931	N/A	N/A	118767	82184	8324	150821
Junyi1564	1564	142	120984	N/A	N/A	86654	34328	116	16389
Junyi2063	2063	3760	290754	N/A	N/A	193664	97090	1432	69050

- DKT [18]: is a pioneer deep learning based knowledge tracing method that uses LSTM to model students' knowledge transition over time.
- DKVMN [23]: is a variant of memory augmented neural networks that model the latent knowledge concept and dynamic student knowledge state over time.
- DeepIRT [21]: is an extension of DKVMN that integrates the one parameter logistic item response theory (1PL-IRT) to provide better interpretability, which could reduce the overfitting issue.
- SAKT [16]: is an attention-based method that leverages the self-attention mechanism to model the interdependencies among interactions on the sequence.
- SAINT [5]: is a transformer-based deep knowledge tracing method, two multi-head attention mechanisms are used to model exercise and response separately.
- AKT [9]: is a variant of transformer-based deep knowledge tracing method that using a monotonic attention mechanism to model the different knowledge transition of students' each historical performance on questions.

In addition to those baselines that support assessed learning materials, we also compare our method with some baselines that either can leverage additional students' non-assessed learning activities by design, or we modify them to consider such non-assessed activities as features of the assessed ones and predict students' future performance. These methods are:

- MLP-M: is a simple multi-layer perceptron that could take query question ID, user ID, and user's 3 past historical records on current query question, as well as 3 most recent non-assessed learning activities as input, and output a probability of user's mastery level on query question.
- DKT-M [24]: is an enhanced DKT model that could incorporate additional question features by concatenating the feature embeddings with exercise response embedding as the input of vanilla DKT.
- SAINT-M [5]: is a variant of SAINT that summing over all embeddings of non-gradable activities along with position encoding as the input of SAINT.
- MVKM [25]: is state of the art method on modeling student knowledge transition over multiple learning resource types based on multiview tensor factorization.

Inspired by the DKT-M [24], we apply the same strategy to DKVMN to incorporate additional non-assessed learning activities as features to end up with method DKVMN-M. Also, inspired by the way of SAINT-M [5] to incorporate rich features into transformer-based model, we apply same strategy as described in the paper into SAKT and AKT to incorporate additional non-assessed learning activities as response features that ends up with baseline methods SAKT-M and AKT-M, respectively.

4.3 Implementation Details

For binary response datasets, including EdNet and Junyi datasets, we convert the response tuple (q_t, r_t) into a single value $z = q_t + r_t \times Q \in \{1, \cdots 2Q\}$ as the lookup key of embedding layer. For numerical response MORF dataset, we feed the tuple (q_t, r_t) into a linear layer to get the embedding. For the question q_t and non-assessed learning resource l_t, we feed their ID into the embedding layers.

For evaluation purpose, we perform the 5-fold user stratified cross-validation for all models and all datasets. Hence, for each fold, 60% users are used as the training set, 20% are validation set, and the rest 20% as test set. For each fold and every method, we use the validation set to tune the hyper-parameters and record the optimal training loss as the condition of early stopping.

We utilize the Gaussian distribution with 0 mean and 0.2 standard deviation to initialize the values of $\mathbf{M}^k$ and $\mathbf{M}_0^v$. We learn the model using the Adam optimization with a learning rate of 0.01 and reduce the learning rate by half once the training loss increases, with the minimal learning $1e$-5 for all methods in 200 max epochs. We also utilize the norm clipping threshold to 50.0 to avoid gradient exploding for all methods. In addition, we follow the general processing steps for knowledge tracing that truncate long sequence and pad short sequence with 0s. The length of sequence is considered as a hyper-parameter of all models which needs to tune. In addition, we also tune the max sequence length of non-assessed learning activities between two assessed learning activities $\mathcal{L}_t$. If the length of non-assessed learning activities is over the maximum size, then we take the most recent ones. Similarly, if the length is less than the required sequence length, we pad with 0s. The table 2 shows the best hyper-parameters of our DMKT on 4 datasets.

We implement the models using PyTorch on a computer with a single NVIDIA Tesla-K80 GPU. For DKT and DKT-M, our implementation is different from the original paper [18], and we follow the same idea suggested by [23] that use norm clipping and early stopping, which could ease the gradient exploding as well as overfitting issues of LSTM. Xavier

Table 2: Hyperparameters of DMKT

Dataset	d_h	N	seq. len.	$\lvert\mathcal{L}_t\rvert$
MORF	128	8	50	8
EdNet	128	8	50	2
Junyi1564	256	8	50	2
Junyi2063	256	32	50	2

initialization is also used to initialize the parameters in DKT and DKT-M. All the baseline methods are implemented in PyTorch and tested to achieve similar performance as reported in the original paper except the SAINT ans SAINT-M. For SAINT, we borrow the implementation from github[6] and extend it to the SAINT-M, since the authors did not release the code.

4.4 Student Performance Prediction

The results of predicting students' performance in the assessed learning resources, including their 95-percentile confidence intervals, are shown in Table 3. The RMSE is measured to evaluate the prediction performance on MORF dataset due to numerical user responses, and the AUC is measured on EdNet and Junyi datasets. A low RMSE score indicates a high prediction performance. An AUC of 0.5 represents the model's performance is equivalent to a random guess model. A high AUC score accounts for a high prediction performance. As you can see, our proposed method, DMKT, achieves the best performance over all baseline methods on all four datasets. This shows that explicitly modeling non-assessed learning materials, along with the assessed ones, is essential in capturing the variations in student performance data.

We can also see that by simply incorporating the non-assessed activities between two assessed activities as additional input features (the "-M" models) the prediction performance is improved in some methods, such as AKT-M on MORF, EdNet, and Junyi datasets. However, unlike attention-based methods which could learn interaction correlation in a long sequence, this kind of simple integration strategy does not improve and may harm the prediction performance in other methods, such as in DKT-M and DKVMN-M, which tend to summarize past historical records as context embeddings. The reason we believe is this trivial integration of non-assessed activities not only loses a large amount of sequential information to model student knowledge transition over time, but also could introduce more noisiness on the data. We conclude that *simply adding the non-assessed learning activities as features, without modeling them explicitly is not enough and may even harm the prediction performance in some models.*

SAINT and SAINT-M have transformer based architecture, which can stack multiple encoders and decoders. However, in our EdNet dataset that contains only $1,000$ users with $200,931$ records on $11,249$ questions, and without additional constraints or regularization as proposed in AKT (another transformer based model), SAINT and SAINT-M can easily overfit the data. MVKM is the only existing baseline method that can explicitly model multiple learning resource types. We can see that it can outperform the deep knowledge tracing methods that uses non-assessed learning materials as features in MORF, which is a mid-size datasets. However, it cannot efficiently run in the larger datasets as the memory usage and linear time complexity over number of interaction records in MVKM limits its applicability on large datasets, such as EdNet and Junyi. Therefore, due to long running time on EdNet and Junyi datasets, we only report its performance in the MORF dataset.

It is worth noting that when our model is fed with assessed learning resources only, it will be equivalent to DKVMN. However, as presented in the table, our proposed model DMKT achieves a better performance over DKVMN as well as DKVMN-M, because DMKT explicitly models the student knowledge transition on non-assessed learning activities, which provides a more accurate encoded information to make the predictions accurately.

4.5 Student Knowledge State Visualization

To see how intractable the discovered student knowledge states are, we visualize the students' knowledge states. Basically, knowledge state visualization shows student's knowledge mastery level on each concept before each attempt on a non-assessed or an assessed learning activity. This provides a useful tool to monitor student knowledge coverage over different concepts and helps instructors to analyze the student's lacking concepts so as to provide tailored instructions for each student. To visualize student knowledge states, we follow the steps in section 3.8 to calculate knowledge state values over all concepts across the student sequence for each student. We show visualization of one example student's knowledge states in the MORF dataset in figure 2. As you can see in the figure, the top x-ticks are labeled with student learning activities. Assessed learning materials (assignments) start with A and non-assessed ones (lecture videos) are annotated by the week they are scheduled and the sequence of video lecture within the week. For example W4V0 means the student has watched week 4 video lecture 0 and A1B denotes the Assignment-1B in week 1. The bottom x-ticks are labeled by either student performance (grade) in the assessed learning materials, or an icon indicating the non-assessed learning resource type. Each row represents one latent concept. In the figure, this student starts with a randomly initialized value memory matrix $\mathbf{M}_0^t$ at time step 0 before working on $A1B$. After finishing the A1B, student's knowledge is updated and increased a little on concept 3 and 6 before working on A3. Student's knowledge grows gradually by working on assignments $A3$ and watching video lectures in week 4. However, student's knowledge drops a little before working on assignment $A4$ and it explains the reason why that student only receives a score 0.3 at the first attempt. Student's knowledge on all concepts grow by working on the assignments until the student started watching video lecture $W6V1$. We can see a slight drop in student's knowledge of some of the concepts (e.g., 7) and increase in other concepts (e.g., 1) while they are watching these videos. One potential reason for the decrease on concept 7 could be the lack of practice with assignments. Watching video lectures indeed improve student knowledge on concept 1 and 2. Another reason for the drop in concept 7 could be related to the student's problem solving ability which results

[6] `https://github.com/Shivanandmn/Knowledge-Tracing-SAINT`

Table 3: Student Performance Prediction Results on 3 Real-World Datasets. Root Mean Square Error (RMSE) and Area Under Curve (AUC) are used to evaluate performance on datasets with numerical feedback and binary feedback, respectively. The average performance over 5 folds as well as 95% confidence interval are reported.

Methods	MORF RMSE	EdNet AUC	Junyi1564 AUC	Junyi2063 AUC
DKT	0.1870 ± 0.0191	0.6393 ± 0.0137	0.8877 ± 0.0050	0.8635 ± 0.0059
DKVMN	0.2042 ± 0.0136	0.6296 ± 0.0104	0.8843 ± 0.0065	0.8558 ± 0.0068
DeepIRT	0.1946 ± 0.0080	0.6290 ± 0.0105	0.8749 ± 0.0053	0.8498 ± 0.0069
SAKT	0.2113 ± 0.0275	0.6334 ± 0.0125	0.8623 ± 0.0047	0.8053 ± 0.0075
SAINT	0.2019 ± 0.0077	0.5205 ± 0.0064	0.8454 ± 0.0096	0.7951 ± 0.0119
AKT	0.2420 ± 0.0155	0.6393 ± 0.0104	0.8311 ± 0.0102	0.8093 ± 0.0091
MVKM	0.1936 ± 0.0096	–	–	–
MLP-M	0.2433 ± 0.0350	0.6102 ± 0.0088	0.7055 ± 0.0191	0.7290 ± 0.0150
DKT-M	0.1927 ± 0.0194	0.6372 ± 0.0120	0.8885 ± 0.0048	0.8652 ± 0.0069
DKVMN-M	0.2251 ± 0.0128	0.6343 ± 0.0074	0.8948 ± 0.0054	0.8513 ± 0.0059
SAKT-M	0.2084 ± 0.0272	0.6323 ± 0.0109	0.8305 ± 0.0071	0.7911 ± 0.0107
SAINT-M	0.1977 ± 0.0055	0.5491 ± 0.0068	0.8454 ± 0.0096	0.7741 ± 0.0139
AKT-M	0.2239 ± 0.0151	0.6404 ± 0.0067	0.8296 ± 0.0093	0.8099 ± 0.0098
DMKT	$\mathbf{0.1369 \pm 0.0195}$	$\mathbf{0.6675 \pm 0.0082}$	$\mathbf{0.9440 \pm 0.0061}$	$\mathbf{0.8714 \pm 0.0069}$

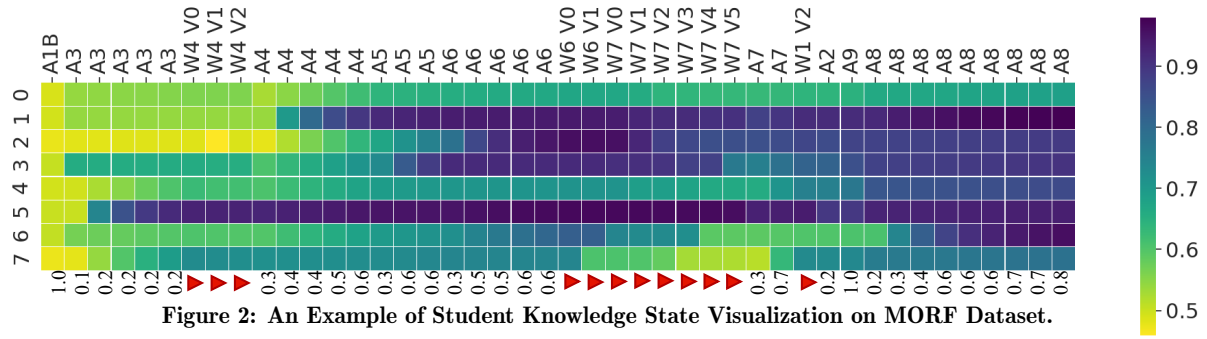


Figure 2: An Example of Student Knowledge State Visualization on MORF Dataset.

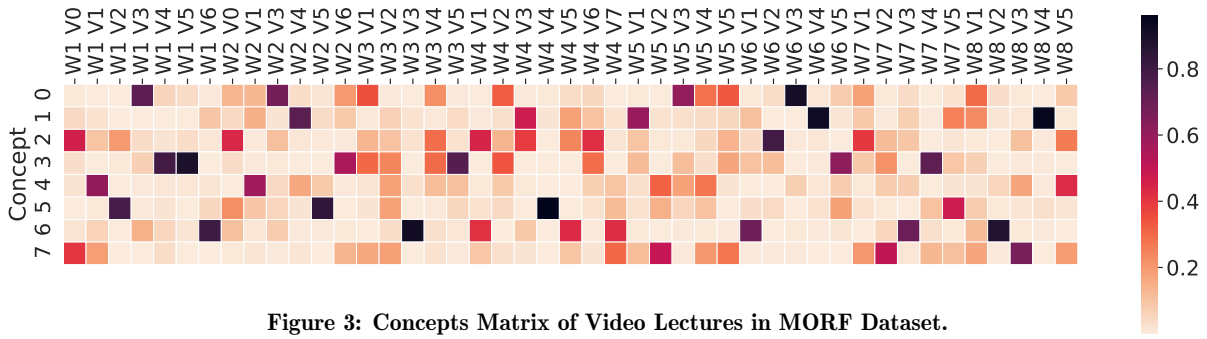
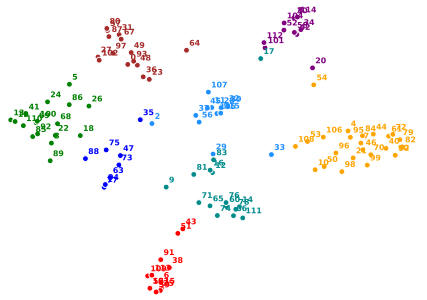


Figure 3: Concepts Matrix of Video Lectures in MORF Dataset.

in their first attempt on Assignment $A7$ to have a score of 0.3. Once the student's first attempt on $A7$ is done, this student quickly masters concept 7 again and their knowledge on all concepts continues to grow along different activities. In this example, it seems *the assessed learning material improves student knowledge more than watching video lectures, which is inline with the previous literature* [10, 13]. Another observation is that this student skips watching video lectures in weeks 1, 2, and 3 before working on assignment $A3$. Similarly, they did not watch videos in week 5 and 6 before trying $A5$ and $A6$. This may explain that this student is not interested in watching video lectures and may not be fully present during watching video lectures which results in tiny knowledge growth over watching them.

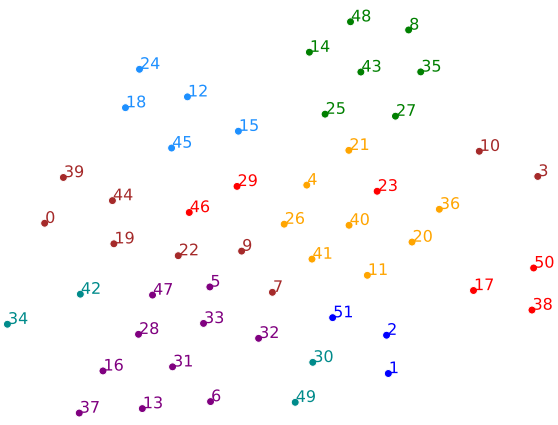
4.6 Concept Discovery

In addition to tracing student knowledge over various types of learning activities, DMKT can provide a feasible solution to discovering underlying patterns or concepts for both assessed and non-assessed learning resources. In other words, the correlation weights $\mathbf{w}$ and $\mathbf{w}^l$, can be interpreted as the importance of latent concepts in each assessed, and non-assessed learning activity respectively. Meaning that, since the key matrix $\mathbf{M}^k$ is used to model the knowledge concepts on the full course, the correlation weight between the learning resources and the concepts implies the strength of



6 telling_time--analog 15 subtraction_1--0 25 subtraction_2--subtraction-2 38 adding_and_subtracting_negative_numbers--MINUS 43 limits_1--3 51 same_side_interior_angles--0 55 reading_tables_1--0 59 measuring_segments--0 91 measuring_and_comparing_angles--compare_4 103 logical_arguments_deductive_reasoning--detachment 109 independent_probability--0 113 conditional_statements_and_truth_value--deduce	4 telling_time_0.5--analog 7 divisibility_intuition--0 10 conditional_statements_2--category 21 addition_2--0 40 prime_numbers--0 44 limits_1--2 46 limits_1--1 50 angle_types--right 53 counting_1--page-numbers 54 counting_1--slices 61 square_roots--square 62 similar_triangles_1--similar 70 converse_inverse_and_contrapositive--equations-and-equations 72 vertical_angles--0 79 reading_line_charts_1--0 82 creating_bar_charts_1--0 84 linear_equations_2--0 90 midpoint_of_a_segment--0 95 converse_inverse_and_contrapositive--elephants-and-peanuts 96 converse_inverse_and_contrapositive--vertical-and-congruent-angles 98 surface_area_of_rectangular_boxes--int-box-T 99 conditional_statements_2--event 106 limits_2--n-lt-d 108 derivative_intuition--polynomial	1 number_line--0 35 expressing_ratios_as_fractions--0 47 limits_2--a-o-xmk 63 conditional_statements_2--adjective 73 complementary_angles--find-hypotenuse 75 converse_inverse_and_contrapositive--sun-and-hat 77 reading_stem_and_leaf_plots--less-than 88 lhopitals_rule--polynomial 94 radius_and_diameter--circle-r-to-d
0 representing_numbers--0 23 number_line_2--where-is-point 27 adding_negative_numbers--PLUS 31 axis_of_symmetry--0 36 congruency_postulates--0 48 evaluating_expressions_1--0 49 angle_types--obtuse 57 reading_tables_2--0 64 congruent_segments--0 67 points_lines_and_planes--line-name 80 reading_bar_charts_2--reading 87 areas_of_kites--kite 93 measuring_and_comparing_angles--measure_2 97 quotient_rule--0 102 logical_arguments_deductive_reasoning--syllogism		2 addition_1--0 11 area_of_parallelograms--par-bh-to-K 28 combining_like_terms_1--none 29 exponents_2--0 32 perimeter_of_squares_and_rectangles--rectangle-lw-to-P 33 decimals_on_the_number_line_2--0 37 ordering_negative_numbers--0 39 adding_and_subtracting_negative_numbers--PLUS 45 limits_1--0 56 exploring_mean_and_median--0 105 limits_2--n-gt-d 107 derivatives_1--quadratic 115 sides_and_angles_of_simple_shapes--polygonshape_vertex
8 comparing_absolute_values--0 20 recognizing_rays_lines_and_line_segments--line-segment 30 telling_time_2--set-hands 34 multiplication_2--0 42 multiplying_decimals--0 52 counting_1--fences 58 reading_pictographs_1--read-chart 101 plotting_the_line_of_best_fit--linear 104 reading_bar_charts_3--midrange 112 inverse_trig_functions--0 114 variance--population	9 converse_inverse_and_contrapositive--dessert-and-pie 12 exponents_1--0 14 multiplication_1--0 16 dividing_decimals_0.5--decimal 17 pythagorean_theorem_1--find-a-leg 60 triangle_types--knownSides 65 measuring_angles--0 66 points_lines_and_planes--name-plane 71 multiplying_and_dividing_scientific_notation--divide 74 converse_inverse_and_contrapositive--rain-and-soccer-practice 76 reading_stem_and_leaf_plots--greater-than 78 reading_bar_charts_1--0 81 reading_bar_charts_2--most-improved 83 linear_equations_1--0 111 triangle_angles_sum--scalene	3 multiplication_0.5--0 5 absolute_value--0 13 division_0.5--0 18 recognizing_rays_lines_and_line_segments--line 19 recognizing_rays_lines_and_line_segments--ray 22 number_line_2--what-is-point 24 number_line_3--0 26 triangle_types--knownAngles 41 composite_numbers--0 68 angle_types--acute 69 combining_like_terms_2--none 85 radians_to_degrees--1 86 degrees_to_radians--degrees-to-radians 89 special_derivatives--0 92 measuring_and_comparing_angles--measure_w_90 100 counting_2--every-n 110 creating_box_and_whisker_plots--0

Figure 4: Cluster Graph of Non-gradable Learning Materials (Hints) in Junyi1564 Dataset Using t-SNE. The question name corresponding to each hint is shown in the right table. (Best viewed in color)



8 W3 V3: Feature Engineering 14 W1 V6: Case Study - San Pedro 25 W4 V5: Advanced BKT 27 W4 V7: Other Structures 35 W6 V1: Learning Curves 43 W7 V3: Advanced Clustering Algorithms 48 W8 V2: Discovery with Models - Case Study	12 W1 V2: Regressors 15 W2 V0: Intro Video 18 W2 V5: Detector Confidence, part 4 24 W4 V4: Item Response Theory 45 W7 V5: Factor Analysis
0 W1 V4: Classifiers Part 2 - RapidMiner 5.3 3 W1 V5: Classifiers Part 3 7 W3 V2: Data Synchronization and Grain Sizes 9 W3 V4: Automated Feature Generation 10 W3 V5: Knowledge Engineering 19 W2 V6: Types of validity 22 W4 V2: Bayesian Knowledge Tracing 39 W6 V5: Other Awesome EDM Visualizations 44 W7 V4: Clustering Examples	30 W5 V2: Casual Mining 34 W6 V0: Intro Video 42 W7 V2: Validation and Selection of K 49 W8 V3: Text Mining
5 W3 V1: Ground Truth for Behavior Detection 6 W3 V0: Intro Video 13 W1 V3: Classifiers Part 1 16 W2 V3: Diagnostic Metrics, part 2 28 W5 V0: Intro Video 31 W5 V3: Association Rule Mining 32 W5 V4: Sequential Pattern Mining 33 W5 V5: Network Analysis 37 W6 V3: Scatter Plots 47 W8 V1: Discovery with Models	4 W2 V2: Diagnostic Metrics, part 1 11 W1 V0 Introduction 20 W4 V0: Intro Video 21 W4 V1: Knowledge Inference 26 W4 V6: Q Matrix 36 W6 V2: Moment by Moment Learning Graphs 40 W7 V0: Intro Video 41 W7 V1: Clustering
	17 W2 V4: Detector Confidence, part 3 23 W4 V3: Performance Factors Analysis 29 W5 V1: Correlation Mining 38 W6 V4: State Space Diagrams 46 W8 V0: Intro Video 50 W8 V4: Hidden Markov Models
	1 W2 V1: Detector Confidence 2 W1 V1: Big Data in Education 51 W8 V5: Conclusions and Further Directions

Figure 5: Cluster Graph of Video Lectures using t-SNE and Titles of Video Lecture of MORF Dataset. Lectures under the same concept are labeled in the same color in the left picture and also are put in the same block in the right table. (Best viewed in color)

their inner relationship. *Not only we can use the correlation weight as latent concepts, we can also use them to find similar learning resources by clustering them over these correlation weights.*

For example, in Figure 3, we visualize the importance of each concept in each of the MORF dataset video lectures. The X-axis ticks show the video lecture weeks and numbers and the Y-axis shows the latent concepts. *As we can see, the concept matrix is relatively sparse, showing that most video lectures strongly belong to 2-3 concepts, while they do have a soft memberships in other concepts too. Many video lectures in the same week have similar concept structures.* For example videos 3, 4, and 5 of week 5 all have a strong representation of concept 0 and videos 0, 1, and 2 of week 1 all are having high correlation weights with concept 2. Given that the course schedule is designed by the instructor, such similarities between the concepts in videos of the same week are expected. *Another interesting observation is the strong appearance of some concepts in videos of different weeks.* For example, concept 1 can be seen in both video 4 of week 8 and video 4 of week 6. This shows that these two video lec-

tures share some similarities that are not represented in class schedule. Looking at the video titles from this course (right-hand side of Figure 5) we can see that the video titles are *State Space Diagrams* and *Hidden Markov Models*, respectively, which are two very closely-related topics To better understand such similarities, we look at grouping of videos according to their discovered concepts in the following.

To this end, we follow the clustering procedures as in [23] to group the learning materials according to the discovered latent concepts. At the same time, we compare these groupings by looking at the problem name associated with each hints and lecture titles for Junyi1564 and MORF datasets in Figures 4 and 5, respectively. To do the clustering, we first assign each learning resource with the concept ID that contains the largest correlation weight as the cluster label. Since there are 8 concepts in total, it results in 8 clusters. Then, we use t-SNE to visualize the clusters, which are shown in the left sides of Figures 4 and 5 for Junyi1564 and MORF datasets, respectively.

As we can see, the resulting t-SNE clusters are more distinct in the Junyi1564 dataset compared to MORF. *In other words, most of clusters in Junyi1564 dataset could be easily separated and distinguished. This implies that the discovered concept matrix of the Junyi1564 dataset is more sparse than the one from the MORF dataset, leading to more outstanding clusters than in MORF, as shown in Figure 4.* Indeed we have seen from Figure 3 that each video lecture in MORF is associated with two to three latent concepts rather than having only one distinct concept. This finding matches these datasets' properties: in the MORF dataset, each assessed learning material is a full complex course problem set which is assigned to students every week, and each non-assessed learning resource is a video lecture that covers multiple knowledge concepts. On the contrary, the assessed learning materials in Junyi1564 dataset are simple math problems, with close-to atomic concept coverage, and the non-assessed resources are hints associated with these problems.

As another result of this clustering, and similar to our findings in Figure 3, we can see that *the more similar or related non-assessed learning materials are clustered together*. For example, in Figure 5, video lectures from week 5 are clustered together, showcasing the similarity between latent concepts in video lectures that are scheduled to be presented together in week 5 of the course. Additionally, video lectures that are conceptually similar to each other can be found grouped together. For example, video lectures from week 6 (*V4 - State Space Diagrams*) and week 8 (*V4 - Hidden Markov Models*), from week 1 (*V2 - Regressors*) and week 7 (*V5 - Factor Analysis*), and from week 1 (V6 - Case Study - San Pedro) and week 8 (*V2 - Case Study - Discovery with models*) are grouped together which are conceptually similar.

These findings are also in accordance with the previous findings in the literature on the MORF dataset [25] and show that DMKT can efficiently discover the underlying concepts presented in the non-assessed learning materials, even though student performance on them is not observable.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed DMKT, the first deep learning based knowledge tracing model that can model and trace student knowledge in both assessed and non-assessed learning resources, find the underlying connects and similarities between learning resources, and predict student performance in the assessed ones. We evaluated DMKT extensively, on four real world datasets and demonstrated that because of its explicit modeling of non-assessed learning materials, its ability in representing non-linear relationships and its capacity in handling larger amounts of data, it outperforms all the baselines, in accurately predicting student performance. We further showcased DMKT's ability in meaningfully tracing student knowledge over assessed and non-assessed learning resources, and the potential effect that each of them can have on student knowledge. In our particular example, we showed that solving problems is a more effective way to learn for our selected student, compared to watching video lectures. Finally, we presented that DMKT can find interpretable latent concepts of non-assessed learning materials, that can be used to group them into meaningful clusters. In the future work, we would like to explore this model on various of learning activities to learn hidden patterns on different learning resources so as to provide tailored learning resource recommendations.

Acknowledgements. This paper is based upon work supported by the National Science Foundation under Grant No. 1755910.

6. REFERENCES

[1] R. Agrawal, M. Christoforaki, S. Gollapudi, A. Kannan, K. Kenthapadi, and A. Swaminathan. Mining videos from the web for electronic textbooks. In *Proceedings of the 12th International Conference on Formal Concept Analysis*, pages 219–234, Berlin, Heidelberg, 2014. Springer.

[2] R. Agrawal, S. Gollapudi, A. Kannan, and K. Kenthapadi. Enriching textbooks with images. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 1847–1856, New York, NY, USA, 2011. ACM.

[3] J. M. L. Andres, R. S. Baker, G. Siemens, D. Gašević, and C. A. Spann. Replicating 21 findings on student success in online learning. *Technology, Instruction, Cognition, and Learning*, pages 313–333, 2016.

[4] J. E. Beck, K.-m. Chang, J. Mostow, and A. Corbett. Does help help? introducing the bayesian evaluation and assessment methodology. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, pages 383–394, Berlin, Heidelberg, 2008. Springer.

[5] Y. Choi, Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and J. Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the 7th ACM Conference on Learning at Scale*, pages 341–344, New York, NY, USA, 2020. ACM.

[6] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1994.

[7] T.-N. Doan and S. Sahebi. Rank-based tensor factorization for student performance prediction. In *Proceedings of the 12th International Conference on Educational Data Mining*. International Educational Data Mining Society, 2019.

[8] F. Drasgow and C. L. Hulin. Item response theory. *Handbook of Industrial and Organizational Psychology*, pages 577–636, 1990.

[9] A. Ghosh, N. Heffernan, and A. S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2330–2339, New York, NY, USA, 2020. ACM.

[10] R. Hosseini, T. Sirkiä, J. Guerra, P. Brusilovsky, and L. Malmi. Animated examples as practice content in a java programming course. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 540–545, New York, NY, USA, 2016. ACM.

[11] Y. Huang, J. P. González-Brenes, and P. Brusilovsky. Challenges of using observational data to determine the importance of example usage. In *Proceedings of the 17th International Conference on Artificial Intelligence in Education*, pages 633–637, Berlin, Heidelberg, 2015. Springer.

[12] H. Khosravi, G. Demartini, S. Sadiq, and D. Gasevic. Charting the design and analytics agenda of learnersourcing systems. In *Proceedings of the 11th International Learning Analytics and Knowledge Conference*, page 32–42, New York, NY, USA, 2021. ACM.

[13] K. R. Koedinger, J. Kim, J. Z. Jia, E. A. McLaughlin, and N. L. Bier. Learning is not a spectator sport: Doing is better than watching for learning from a mooc. In *Proceedings of the 2nd ACM Conference on Learning at Scale*, page 111–120, New York, NY, USA, 2015. ACM.

[14] A. S. Lan, A. E. Waters, C. Studer, and R. G. Baraniuk. Sparse factor analysis for learning and content analytics. *The Journal of Machine Learning Research*, 15(1):1959–2008, 2014.

[15] A. S. Najar, A. Mitrovic, and B. M. McLaren. Adaptive support versus alternating worked examples and tutored problems: Which leads to better learning? In *Proceedings of the 22nd International Conference on User Modeling, Adaptation, and Personalization*, pages 171–182, Berlin, Heidelberg, 2014. Springer.

[16] S. Pandey and G. Karypis. A self-attentive model for knowledge tracing. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 384–389. International Educational Data Mining Society, 2019.

[17] R. Pelánek. Measuring similarity of educational items: An overview. *IEEE Transactions on Learning Technologies*, 13(2):354–366, 2019.

[18] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, page 505–513, Cambridge, MA, USA, 2015. MIT Press.

[19] S. Sahebi and P. Brusilovsky. Student performance prediction by discovering inter-activity relations. *International Educational Data Mining Society*, 2018.

[20] S. Sahebi, Y.-R. Lin, and P. Brusilovsky. Tensor factorization for student modeling and performance prediction in unstructured domain. *International Educational Data Mining Society*, 2016.

[21] C. K. Yeung. Deep-irt: Make deep learning based knowledge tracing explainable using item response theory. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 683–686. International Educational Data Mining Society, 2019.

[22] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *Proceedings of the 18th International Conference on Artificial Intelligence in Education*, pages 171–180, Berlin, Heidelberg, 2013. Springer.

[23] J. Zhang, X. Shi, I. King, and D.-Y. Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web*, pages 765–774, New York, NY, USA, 2017. ACM.

[24] L. Zhang, X. Xiong, S. Zhao, A. Botelho, and N. T. Heffernan. Incorporating rich features into deep knowledge tracing. In *Proceedings of the 4th ACM Conference on Learning at Scale*, pages 169–172, New York, NY, USA, 2017. ACM.

[25] S. Zhao, C. Wang, and S. Sahebi. Modeling knowledge acquisition from multiple learning resource types. In *Proceedings of The 13th International Conference on Educational Data Mining*, pages 313–324. International Educational Data Mining Society, 2020.

The Effect of an Intelligent Tutor on Performance on Specific Posttest Problems

Adam Sales
Worceseter Polytechnic Institute
asales@wpi.edu

Ethan Prihar
Worceseter Polytechnic Institute
ebprihar@gmail.com

Neil Heffernan
Worceseter Polytechnic Institute
nth@wpi.edu

John F. Pane
Rand Corporation
jpane@rand.org

ABSTRACT

This paper drills deeper into the documented effects of the Cognitive Tutor Algebra I and ASSISTments intelligent tutoring systems by estimating their effects on specific problems. We start by describing a multilevel Rasch-type model that facilitates testing for differences in the effects between problems and precise problem-specific effect estimation without the need for multiple comparisons corrections. We find that the effects of both intelligent tutors vary between problems– the effects are positive for some, negative for others, and undeterminable for the rest. Next we explore hypotheses explaining why effects might be larger for some problems than for others. In the case of ASSISTments, there is no evidence that problems that are more closely related to students' work in the tutor displayed larger treatment effects.

Keywords

Causal impact estimates,multilevel modeling,intelligent tutoring systems

1. INTRODUCTION: AVERAGE AND ITEM-SPECIFIC EFFECTS

The past decade has seen increasing evidence of the effectiveness of intelligent tutoring systems (ITS) in supporting student learning [7][13]. However, surprisingly little detail is known about these effects such as which students experience the biggest benefits, under what conditions. This paper will focus on the question of which areas of learning had the largest impact in two different year-long randomized trials: of the Cognitive Tutor Algebra I curriculum (CTA1) [17] and of the ASSISTments ITS [22].

Large-scale efficacy or effectiveness trials in education research, including evaluations of ITS [17][18][22], often estimate the effect of an educational intervention on student scores on a standardized test. These tests consist of many items, each of which tests student abilities in, potentially, a separate set of skills. Prior to estimating program effects, analysts collapse data across items into student scores, often using item response theory models [25] that measure both item- and student-level parameters. Then, these student scores are compared between students assigned to the intervention group and those assigned to control.

This approach has its advantages, in terms of simplicity and (at least after aggregating item data into test scores) model-free causal identification. If each item is a measurement of one underlying latent construct (such as "algebra ability") aggregating items into test scores yields efficiency gains. However, in the (quite plausible) case that posttest items actually measure different skills, and the impact of the ITS varies from skill to skill, item-specific impacts can be quite informative.

In the case of CTA1 and ASSISTments, we find that, indeed, the ITS affect student performance differently on different posttest items, though at this stage it is unclear why the affects differed.

The following section gives an overview of the two large-scale ITS evaluations we will discuss, including a discussion of the available data and of the two posttests. Next, Section 3 will discuss the Bayesian multilevel model we use to estimate item-specific effects, including a discussion of multiple comparisons; Section 4 will discuss the results—estimates of how the two ITS impacted different posttest items differently; Section 5 will present a preliminary exploration of some hypotheses as to why ASSISTments may have impacted different skills differently; and Section 6 will conclude.

2. THE CTA1 AND ASSISTMENTS TRIALS

This paper uses data from two large-scale field trials of ITSs CTA1 and ASSISTments. The CTA1 intervention consisted of a complete curriculum, combining the Cognitive Tutor ITS, along with a student-centered classroom curriculum. CTA1 was a created and run by Carnegie Learning; an up-

Adam Sales, Ethan Prihar, Neil Heffernan and John Pane "The Effect of an Intelligent Tutor on Performance on Specific Posttest Problems". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 206-215. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

dated version of the ITS is now known as Mathia. The Cognitive Tutor is described in more detail in [2] and elsewhere, and the effectiveness trial is described in [17]. ASSISTments is a free online-homework platform, hosted by Worcester Polytechnic Institute, that combines electronic versions of textbook problems, including on-demand hints and immediate feedback, with bespoke mastery-based problem sets known as "skill builders." ASSISTments is described in [10] and the efficacy trial is described in [22].

This section describes the essential aspects of the field trials and the data that we will use in the rest of the paper.

2.1 The CTA1 Effectiveness Trial

From 2007 to 2010, the RAND Corporation conducted a randomized controlled trial to compare the effectiveness of the CTA1 curriculum to business as usual (BaU). The study tested CTA1 under authentic, natural conditions, i.e., oversight and support of CTA1's use was the same as it would have been if there was not a study being conducted. Nearly 20,000 students in 70 high schools ($n = 13,316$ students) and 76 middle schools ($n = 5,938$) located in 52 diverse school districts in seven states participated in the study. Participating students in Algebra I classrooms took an algebra I pretest and a posttest, both from the CTB/McGraw-Hill Acuity series.

Schools were blocked into pairs prior to randomization, based on a set baseline, school-level covariates, and within each pair, one school was assigned to the CTA1 arm and the other to BaU. In the treatment schools, students taking algebra I were supposed to use the CTA1 curriculum, including the Cognitive Tutor software; of course, the extent of compliance varied widely [12][11].

Results from the first and second year of the study were reported separately for middle and high schools. In the first year, the estimated treatment effect was close to zero in middle schools and slightly negative in high schools. However, the 95% confidence intervals for both these results included negative, null, and positive effects. In the second year, the estimated treatment effect was positive–roughly one fifth of a standard deviation—for both middle and high schools, but it was only statistically significant in the high school stratum.

In this study, we make use of students' overall scores on the pretest, anonymized student, teacher, school, and randomization block IDs, and an indicator variable for whether each student's school was assigned to the CTA1 or BaU, along with item-level posttest data: whether each student answered each posttest item correctly. For the purposes of this study, skipped items were considered incorrect.

2.1.1 Posttest: The Algebra Proficiency Exam

The RAND CTA1 study measured the algebra I learning over the course of the year using the McGraw-Hill Algebra Proficiency Exam (APE). This was a multiple choice standardized test with 32 items testing a mix of algebra and pre-algebra skills. Table 1, categorizes the test's items by the algebra skills they require, and gives an example of a problem that would fall into each category. The categorization was taken from the exam's technical report [6].

2.2 The Maine ASSISTments Trial

From 2012–2014, SRI International conducted an randomized field trial in the state of Maine to estimate the efficacy of ASSISTments in improving 7th grade mathematics achievement. Forty-five middle schools from across the state of Maine were randomly assigned between two conditions: 23 middle schools were assigned to a treatment condition; mathematics teachers in these schools were instructed to use ASSISTments to assign homework, receiving support and professional development while doing so. The remaining 22 schools in the BaU condition were barred from using ASSISTments during the course of the study but were offered the same resources and professional development as the treatment group after the study was over. The study was conducted in Maine due to the state's program of providing every student with a laptop, which allowed students to complete homework online.

The 45 participating schools were grouped into 21 pairs and one triplet based on school size and prior state standardized exam scores; one school in each pair, and two schools in the triplet, were assigned to the ASSISTments condition, with the remaining schools assigned to BaU. Subsequent to random assignment, one of the treatment schools dropped out of the study, but its matched pair did not. Although the study team continued to gather data from the now-unmatched control school, that data was not included in the study. However, we are currently unable to identify which of the control schools was excluded from the final data analysis, so the analysis here includes 44 schools, while [22] includes only 43.

The study measured student achievement on the standardized TerraNova math test at the end of the second year of implementation, and estimated a treatment effect of 0.18 ± 0.12 standard deviations.

In this study, we make use of anonymized student, teacher, school, and randomization block IDs, and an indicator variable for whether each student's school was assigned to the ASSISTments or BaU, along with item-level posttest data: whether each student answered each posttest item correctly. For the purposes of this study, skipped items were considered incorrect. The initial evaluation included a number of student-level baseline covariates drawn from Maine's state longitudinal data system, include prior state standardized test scores. We do not currently have access to that data; the only covariate available was an indicator of whether each student was classified as special education.

2.3 The TerraNova Test

The primary outcome of the ASSISTments Maine trial was students' scores on the TerraNova Common Core assessment mathematics test, published by Data Recognition Corporation CTB. The TerraNova assessment includes 37 items, 32 of which were multiple choice and 5 of which were open response. Unfortunately, we detected an anomaly in the item-level data for the open-response questions, so this report will focus only on the 32 multiple choice questions.

The items are supposed to align with the Common Core State Standards, but the research team was not given a document aligning CCSS with the test items. Instead, a

Objective	Items	Example
Functions and Graphs	6, 8, 19, 20, 22, 23, 27, 31, 32	Which of these points is on the graph of [function]
Geometry	12, 18, 24, 29	Find the length of the base of the right triangle shown below
Graphing Linear Equations	5, 9, 15, 17, 26	Which of the lines below is the graph of [linear equation]?
Quadratic Equations and Functions	2, 25, 28, 30	Which of these shows a correct factorization of [quadratic equation]?
Solving Linear Equations and Linear Inequalities	1, 4, 11, 13, 16	Solve the following system of equations
Variables, Expressions, Formulas	3, 7, 10, 14, 21	Which of these expressions is equivalent to the one below?

Table 1: Objectives required for the 32 items of the Algebra Proficiency Exam, the posttest for the CTA1 Evaluation

member of the ASSISTments staff with expertise in middle school education aligned them according to her best judgment. Table 2 gives this alignment. More information on specific standards can be found at the CCSS website [16].

3. METHODOLOGY: MULTILEVEL EFFECTS MODELING

In principal estimating program effects on each posttest item is straightforward: the same model used to estimate effects on student overall scores could be used to estimate effects on each item individually (perhaps—but not necessarily—adapted for a binary response). However, estimating 32 separate models for each stratum of the CTA1 study, and 32 separate models for the ASSISTments study ignores multilevel structure of the dataset, and leads to imprecise estimates. Moreover, doing so invites problems of multiple comparisons—between the four strata of the CTA1 study and the ASSISTments study, there are 160 separate effects to estimate. If each estimate is subjected to a null hypothesis test at level $\alpha = 0.05$, even if neither ITS affected test performance at all, we would still expect to find roughly eight significant effects.

Instead, we estimated item-specific effects with a multilevel logistic regression model model [8], based roughly on the classic "Rasch" model of item response theory [25][20]. That is, we estimated all item-specific effects for a particular experiment simultaneously, with one model, in which the item-specific effect estimates are random effects. The separate effects were modeled as if drawn from a normal distribution with a mean and standard deviation estimated from the data. This normal distribution can be thought of as a Bayesian prior distribution; the fact that its parameters are estimated from the data puts us in the realm of empirical Bayes [5]. This prior distribution acts as a regularizer, shrinking the several item-specific effect estimates towards their mean [15]. Although doing so incurs a small amount of bias, it reduces standard errors considerably while maintaining the nominal coverage of confidence intervals [23].

Gelman, Hill, and Masanao [9] argue that estimating a set of different treatment effects within a multilevel model also obviates the need for multiplicity corrections. Generally speaking, the reason for spurious significant results is that as a group of estimates gets larger, so does the probability that one of them will exceed the test's critical value. In other words, as a the set of estimates grows, so does their maximum (and their minimum, in magnitude). Multilevel modeling helps by shrinking the most extreme estimates towards their common mean. Since extreme values are less likely in a multilevel model, so are spuriously significant effect estimates.

A small simulation study in the Appendix (mostly) supports Gelman et al.'s argument. As the number of estimated effects grows, the familywise error rate (i.e. the probability of *any* type-I error in a group of tests) grows rapidly if effects are estimated and tested separately, but not if they are estimated simultaneously in a multilevel model. However, the error rates for the multilevel model effect estimates are slightly elevated—hovering between 0.05 and 0.075 throughout. There is good reason to believe that a fully Bayesian approach will improve these further (see, e.g., [21], p. 425).

3.1 The Model for the CTA1 Posttest

For the CTA1 RCT, we estimated a separate model for high school and middle school, but we combined outcome data across the two years. Let $Y_{ij} = 1$ if student i answered item j correctly, and let $\pi_{ij} = Pr(Y_{ij} = 1)$. Then the multilevel logistic model was:

$$\begin{aligned} logit(\pi_{ij}) &= \beta_0 + \beta_1 Year2_i + \beta_2 Trt_i + \beta_3 Pretest_i \\ &+ \beta_4 Year2_i Trt_i + \beta_5 Year2_i Pretest_i \\ &+ \gamma_{j0} + \gamma_{j1} Trt_i + \gamma_{j2} Year2_i + \gamma_{j3} Year2_i Trt_i \\ &+ \delta_i + \eta_{cls[i]} + \epsilon_{sch[i]} \end{aligned} \quad (1)$$

Where $Year2_i = 1$ if student i was in the 2nd year of the study and 0 otherwise, $Trt_i = 1$ if student i was in a school assigned to treatment, and $Pretest_i$ is i's pretest score. The coefficients β_0–β_5 are "fixed effects," that is, they are not given any probability model. γ_{j0}–γ_{j3} vary with posttest item j, and are modeled jointly as multivariate normal: $\boldsymbol{\gamma} \sim MVN(\mathbf{0}, \Sigma)$, where Σ is a 4×4 covariance matrix for the γ terms. Similarly, the random intercepts δ_i, $\eta_{cls[i]}$, and $\epsilon_{sch[i]}$, which vary at the student, classroom, and school level, are each modeled as univariate normal with mean 0 and a standard deviation estimated from the data.

Collecting like terms in model (1), note that for a student in the first year of the study, the effect of assignment to the CTA1 condition is $\beta_2 + \gamma_{j1}$ on the logit scale; in other words, the effects of assignment to CTA1 in year 1 are mod-

CCSS	Items
Expressions and Equations	17,28
Functions (8G)	26,27
Geometry	12,16,19,21,23,31
Make sense of problems and persevere in solving them (MP)	13
Ratios and Proportional Relationships	22,24,25,29
Reason abstractly and quantitatively (MP)	15,20
Statistics and Probability	10,11,32
The Number System	1,2,3,4,5,6,7,8,9,14,18

Table 2: Common Core State Standards (CCSS) for the 32 multiple choice TerraNova items, as identified by the ASSISTments team. Standards are from grade 7 except where indicated–grade 8 (8G) or Mathematical Practice (MP)

eled as normal with a mean of β_2 and a variance of Σ_{22}. The variance Σ_{22} estimates the extent to which the effect of assignment to the CTA1 condition varies from one problem to another. If the effect were the same for every posttest problem, we would have $\Sigma_{22} = 0$. For students year 2, the effect on problem j is $\beta_2 + \beta_4 + \gamma_{j1} + \gamma_{j3}$ on the logit scale—the effects are normally distributed with a mean of $\beta_2 + \beta_4$ and a variance of $\Sigma_{22} + \Sigma_{44} + 2\Sigma_{24}$. The Σ matrix also includes the covariance between the effects of the intervention on items in year 1 and the effects on the same items in year 2 as

$$Cov(\gamma_{j1}, \gamma_{j1} + \gamma_{j3}) = Var(\gamma_{j1}) + Cov(\gamma_{j1}, \gamma_{j3}) = \Sigma_{22} + \Sigma_{23}$$

Likelihood ratio tests using the χ^2 distribution can test the null hypothesis that the variance of treatment effects are 0. For simplicity, we did so using separate models for the two years, rather than the combined model (1).

The treatment effects themselves are estimated using the BLUPs (best linear unbiased predictors) for the random effects $\boldsymbol{\gamma}$. In many contexts, random effects are considered nuisance parameters, and primary interest is in the fixed (unmodeled) effects $\boldsymbol{\beta}$. However, there is a long tradition, mostly in the Bayesian and empirical Bayes literature, of using BLUPs for estimation of quantities of interest. The models were fit in `R` [19] using the `lme4` package [3], which provides empirical Bayesian estimates of the conditional (or posterior) variance of the BLUPs, which we use (in combination with the estimated standard errors for fixed effects) in constructing confidence intervals for item-specific effects.

3.2 The Model for the ASSISTments Posttest

The model for estimating item-specific effect of ASSISTments on TerraNova items was highly similar to model (1). There were three important differences: first, there was only one year of data. Second, we did not have access to pretest scores, but we did include an indicator for special education status as a covariate. Lastly, the hierarchical variance structure for student errors was somewhat different—we included an error term for teacher instead of classroom, and included random intercepts for randomization block.[1]

[1]In linear models it is typically recommended to include fixed effects for randomization block [4]. In logistic regression, including a large number of fixed effects violates the assumptions underlying the asymptotic [1]. We tried it both ways and found that it made little difference.

All in all, the model was:

$$\begin{aligned} logit(\pi_{ij}) &= \beta_0 + \beta_1 Trt_i + \beta_2 SpEd_i \\ &+ \gamma_{j0} + \gamma_{j1} Trt_i \\ &+ \delta_i + \eta_{tch[i]} + \epsilon_{sch[i]} + \zeta_{pair[i]} \end{aligned} \quad (2)$$

where $SpEd_i = 1$ if student i is classified as needing special education, $\eta_{tch[i]}$ is a random intercept for i's teacher, and $\zeta_{pair[i]}$ is a random intercept for i's school's randomization block. The rest of the parameters and variables are defined the same as in (1). The treatment effect on problem j is modeled as $\beta_1 + \gamma_{j1}$ for multiple choice items. The random effects $\boldsymbol{\gamma} \sim N(\mathbf{0}, \Sigma)$ where Σ is a 2×2 covariance matrix.

4. MAIN RESULTS: ON WHICH ITEMS DID ITSS BOOST PERFORMANCE?

4.1 CTA1

Figure 1 gives the results from model (1) fit to the middle school and to the high school sample. Each point on the plot represents the estimated effect of assignment to the CTA1 condition on the log odds of a correct answer on one posttest item. The estimates are accompanied by approximate 95% confidence intervals.

It is immediately clear that the effect of assignment to CT vary between posttest items–indeed the χ^2 likelihood ratio test rejects the null hypothesis of no treatment effect variance with $p < 0.001$ in all four strata.

In the middle school sample, the average treatment effect across items was close to 0 for both years (-0.08 in year 1 and 0.03 in year 2 on the logit scale), and not statistically significant. However, the standard deviation of treatment effects between problems was much higher—0.31 in year 1 and 0.29 in year 2, implying that assignment to CTA1 boosted performance on some problems and hurt performance on others. To interpret the standard deviation of effects on the probability scale, consider that for a marginal student, with a 1/2 probability of answering an item correctly, a difference of 0.3 between two treatment effects would correspond to a difference in the probability of a correct answer of about 7.5% (using the "divide by 4 rule" of [8] p. 82). The effects are also moderately correlated across the two years, with $\rho \approx 0.4$—items that CTA1 impacted in year 1 were somewhat likely to be similarly impacted in year 2.

Many of the treatment effects in the upper pane of Figure 1 are estimated with too much noise to draw strong

Figure 1: Estimated treatment effects of CTA1 for each level—high school or middle school—implementation year, and posttest item, with approximate 95% confidence intervals

conclusions—the sample size was substantially smaller in the middle school stratum than in the high school stratum. However, some effects are discernible: in year 1, effects were negative, and on the order of roughly 0.4 on the logit scale (0.1 on the probability scale for a marginal student) on items 1, 2, 9, 10, 12, 19, 22, and 25, and on the order of approximately 0.7 for item 17 (which asks students to match a linear equation to its graph), and similarly-sized positive effects on items 27, 30, and 32. In year 2 there were fewer clearly negative effects—on items 1 and 7—and more positive effects, such as on items 16, 18, 22, 29, and 32. There is a striking difference between the year 1 and year 2 effects on item 22, which asks students to match a quadratic expression to its graph—the effect was quite negative in year 2 and quite positive in year 2.

In the high school sample, the average treatment effect across items was roughly -0.1 in year 1 and 0.13 in year 2, on the logit scale, neither statistically significant–though the difference between the average effect in the two years was significant ($p < 0.001$). The effects varied across items, though less widely in high school than in middle school—in both years the standard deviation of item-specific effects was roughly 0.17. Item-specific effects were more highly correlated across years ($\rho \approx 0.69$)—at some points in the lower pane of Figure 1 it appears as though the curve from year 2 was simply shifted up from year 1.

The item-specific effects in the high school sample were estimated with substantially more precision than in the middle school sample, due to a larger sample size. In year 1, there were striking negative effects on items 2, 14, and 25 which ask students to manipulate algebraic expressions, and on item 12, which ask students to calculate the length of the side of a triangle. In year 2, these negative effects disappeared. Instead, there were positive effects, especially on items 8 and 22, which both ask about graphs of algebraic functions, and on a stretch of items from 15–22. The difference in the estimated effects between years was positive for all items and highest for problems 2, 20, and 25, which ask students to manipulate or interpret algebraic expressions, and 12, the triangle problem. In items 2, 12, and 25, the effect was significantly negative in year 1 and closer to zero in year 2, while for item 20 the effect was close to zero in year 1 and positive in year 2.

Figure 2 plots the estimated effect on each posttest item as a function of the item's objective in Table 1. Some patterns are notable. There was a wide variance in the effects on the four geometry problems for middle schoolers in year 1, but in year 2 all the effects on geometry items were positive and roughly the same size. The geometry items in the high school sample follow a similar, if less extreme, pattern. Across both middle and high school, the largest positive effects were for Functions and Graphs problems, especially item 22 for year 2; on items 23, 27, 31 and 32, middle schoolers—especially in year 2—saw positive effects while high schoolers saw effects near 0.

4.2 ASSISTments

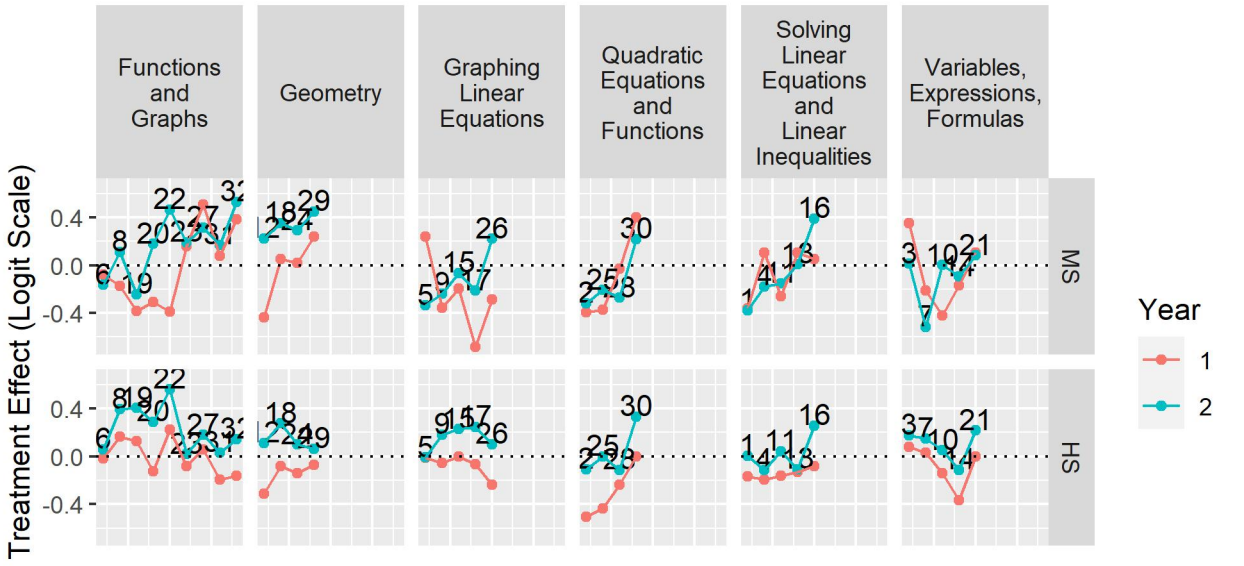


Figure 2: Estimated treatment effects of CTA1 posttest items arranged by the group of skills each item is designed to test. See Table 1 for more detail.

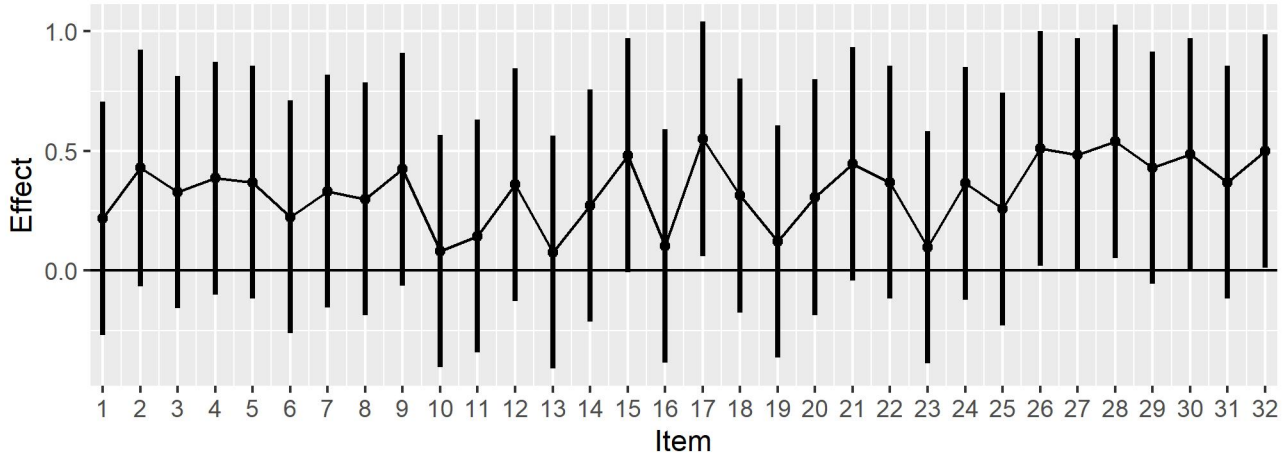


Figure 3: Estimated treatment effects of ASSISTments for each multiple choice posttest item, with approximate 95% confidence intervals

Figure 3 gives the results from model (2), plotting item-specific effect estimates with approximate 95% confidence intervals for each multiple choice TerraNova posttest item. The model estimated an average effect of 0.33, with a standard error of 0.23, for multiple choice problems. The standard deviation of item-specific effects was positive ($p<0.001$) but less than for the CTA1 items: it was estimated as 0.16 on the logit scale. The confidence intervals in Figure 3 are also much wider than those for CTA1; we suspect that a large part of the reason is that we did not have access to pretest scores, an important covariate.

The largest effects on the multiple choice items were 28 and 17, which both required students to plug in values for variables in algebraic expressions. The confidence intervals around the effects for items 26 and 32 also exclude 0.

Figure 4 plots item-specific effects for multiple choice TerraNova items grouped according to their CCSS, as in Table 2, with the non-grade-7 standards grouped together as "Other." Interestingly, the largest effects tended to be for items in this "Other" category—as did the smallest effect, for item 13. Effects for problems in the "Number System" and "Ratios and Proportional Relationships" categories had the most consistent effects, between 0.2 and 0.4 on the logit scale.

5. EXPLORING HYPOTHESES ABOUT *WHY* ASSISTMENTS EFFECTS DIFFERED

Researchers on the ASSISTments team have built on the CCSS links of Table 2, linking TerraNova posttest items to data on student work within ASSISTments, for students in the treatment condition. This gives us an opportunity to use student work within ASSISTments to explain some of the variance in treatment effects.

Like TerraNova items in Table 2, ASSISTments problems are linked with CCSS. By observing which problems treatment students worked on, and using this linkage, we could observe which Common Core standards they worked on the most within ASSISTments. We hypothesized that treatment effects might be largest for the TerraNova problems that were linked with the Common Core standards students spent the most time working on. In other words, we linked TerraNova items with worked ASSISTments problems *via* Common Core standards. The Common Core linkage we used in this segment was finer-grained than Table 2, so TerraNova items in the same category in Table 2 may not be linked with the same problems in this analysis.

We examined our hypothesis in two ways: examining the relationships between treatment effects and the number of related ASSISTments problems students in the treatment group worked, and the number of related ASSISTments problems students in the treatment group worked *correctly*. This analysis includes two important caveats: first, the linkages, both between TerraNova items and CCSS, and between ASSISTments problems and CCSS, were subjective and error-prone, possibly undermining the linkage between TerraNova items and ASSISTments problems. Secondly, student work in ASSISTments is necessarily a post-treatment variable—it was affected by treatment assignment. If the treatment randomization had fallen out differently, different schools would have been assigned to the ASSISTments condition and different ASSISTments problems would have been worked. Including the number of worked or correct related problems as a predictor in a causal model risks undermining causal interpretations [14].

Figures 5 and 6 plot estimated item-specific effects for multiple choice TerraNova items against the number of ASSISTments problems that students in the treatment arm worked or worked correctly, respectively, over the course of the RCT. The X-axis is on the square-root scale, and a loess curve is added for interpretation. Little, if any, relationship is apparent in either figure, suggesting either the lack of a relationship between specific ASSISTments work and posttest items, or issues with the linkage. This is hardly surprising, given both the difficulty in linking ASSISTments and TerraNova problems, and given the fact that topics in mathematics are inherently connected, so that improving one skill tends to improve others as well.

6. CONCLUSIONS

Education researchers are increasingly interested in "what works." However, the effectiveness of an intervention is necessarily multifaceted and complex—effects differ between students, as a function of implementation [24], and, potentially, as a function of time and location. In this paper we explored a different sort of treatment effect heterogeneity—differences in effectiveness for different outcomes—specifically, different posttest items measuring different skills. Collapsing item-level posttest data into a single test score has the advantage of simplicity (which is nothing to scoff at, especially in complex causal scenarios) but at a cost. Analysis using only summary test scores squanders a potentially rich source of variability and information about intervention effectiveness that is already at our fingertips. There is little reason *not* to examine item-specific effects.

In this paper, we showed how to estimate item specific effects using a Bayesian or empirical Bayesian multilevel modeling approach that, we argued, can improve estimation precision and avoid the need for multiplicity corrections. The estimates we provided here combine maximum likelihood estimation and empirical Bayesian inference; there is good reason to suppose that a fully Bayesian approach would provide greater validity, especially in standard error estimation and inference. However, fitting complex multilevel models using Markov Chain Monte Carlo methods is computationally expensive, and can be very slow, even with the latest software. We hope to explore this option more fully in future work.

While estimating item-specific effects is relatively straightforward, interpreting them presents a significant challenge. This is due to a number of factors: first, when looking for trends in treatment effects by problem attributes, the sample size is the number of exam items, not the number of students, so patterns can be hard to observe and verify. Secondly, there is a good deal of ambiguity and subjectivity involved in defining and determining item attributes and features, which is exacerbated by the fact that standardized tests generally cannot be made publicly available. Lastly, since student ITS work over the course of a study is necessarily post-treatment assignment, careful causal modeling (such as principal stratification [24]) may be necessary. Ex-

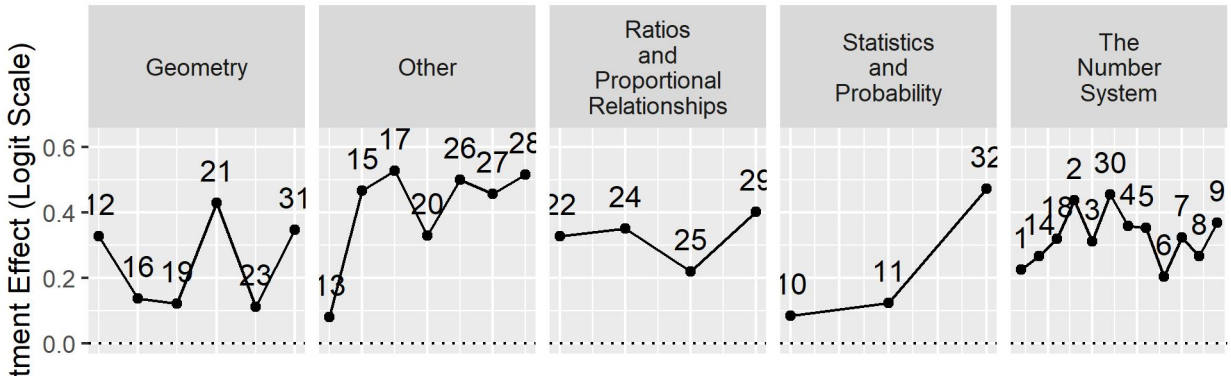


Figure 4: Estimated treatment effects of ASSISTments for each multiple choice posttest item, arranged according to CCSS, as in Table 2. The "Other" category includes Functions and the two Mathematical Practice standards, "make sense of problems and persevere in solving them" and "reason abstractly and quantitatively".

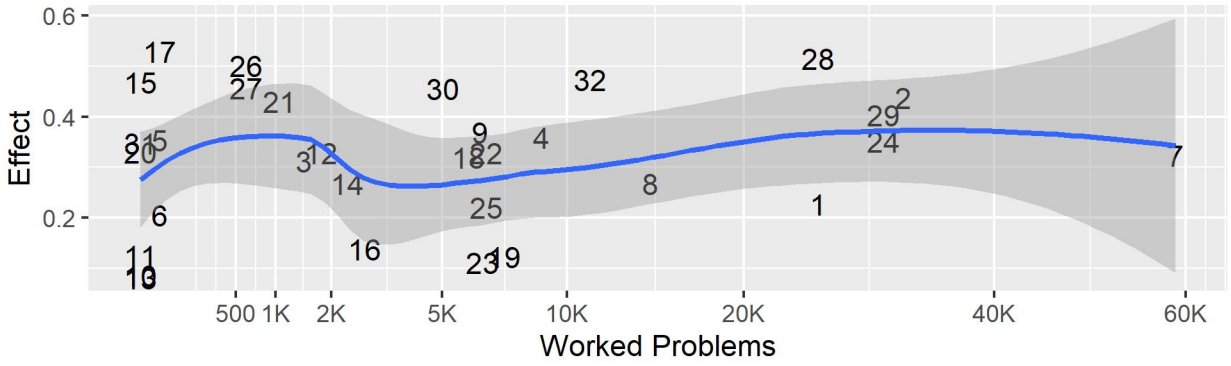


Figure 5: Estimated effects on multiple-choice TerraNova items plotted against the number of related ASSISTments problems that students in the treatment arm worked over the course of the study. The X-axis is plotted on the square-root scale, and a non-parametric loess fit is added for interpretation.

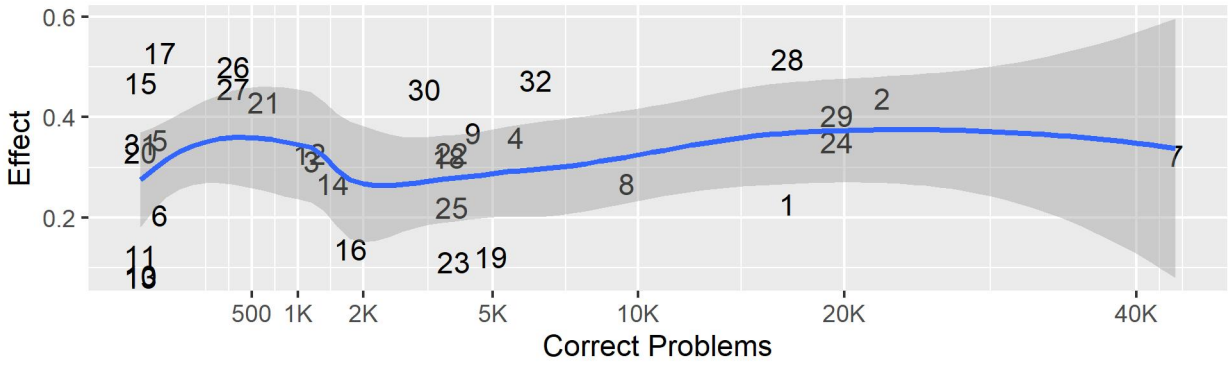


Figure 6: Estimated effects on multiple-choice TerraNova items plotted against the number of related ASSISTments problems that students in the treatment arm worked correctly over the course of the study. The X-axis is plotted on the square-root scale, and a non-parametric loess fit is added for interpretation.

amining heterogeneity between item-specific treatment effects may play a larger role in helping to generate hypotheses about ITS effectiveness than in confirming hypotheses.

Despite those difficulties, the analysis here uncovered important information about the CTA1 and ASSISTments effects. First, the discovery that the effects vary between items is notable in itself. In our analysis of CTA1 we noticed that some of the largest effects—and differences between first and second-year effects— were for posttest items involving manipulating algebraic expressions and interpreting graphs. In our analysis of ASSISTments, we discovered a large difference between negative effects on open-ended questions and positive effects on multiple choice questions, and also that the largest effects were on problems requiring students to plug numbers into algebraic expressions.

We hope that this research will serve as a proof-of-concept and spur further work delving deeper into data we already have.

7. REFERENCES

[1] A. Agresti. *Categorical data analysis*, volume 482. John Wiley & Sons, 2003.

[2] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207, 1995.

[3] D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015.

[4] H. S. Bloom, S. W. Raudenbush, M. J. Weiss, and K. Porter. Using multisite experiments to study cross-site variation in treatment effects: A hybrid approach with fixed intercepts and a random treatment coefficient. *Journal of Research on Educational Effectiveness*, 10(4):817–842, 2017.

[5] G. Casella. An introduction to empirical bayes data analysis. *The American Statistician*, 39(2):83–87, 1985.

[6] CTB/McGraw-Hill. Acuity algebra proficiency technical report. Monterey, CA, 2007.

[7] M. Escueta, V. Quan, A. Nickow, and P. Oreopoulos. Education technology: An evidence-based review. *NBER Working Paper*, (w23744), 2017.

[8] A. Gelman and J. Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press, 2006.

[9] A. Gelman, J. Hill, and M. Yajima. Why we (usually) don't have to worry about multiple comparisons. *Journal of Research on Educational Effectiveness*, 5(2):189–211, 2012.

[10] N. T. Heffernan and C. L. Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4):470–497, 2014.

[11] A. Israni, A. C. Sales, and J. F. Pane. Mastery learning in practice: A (mostly) descriptive analysis of log data from the cognitive tutor algebra i effectiveness trial, 2018.

[12] R. Karam, J. F. Pane, B. A. Griffin, A. Robyn, A. Phillips, and L. Daugherty. Examining the implementation of technology-based blended algebra i curriculum at scale. *Educational Technology Research and Development*, 65(2):399–425, 2017.

[13] J. A. Kulik and J. Fletcher. Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of educational research*, 86(1):42–78, 2016.

[14] J. M. Montgomery, B. Nyhan, and M. Torres. How conditioning on posttreatment variables can ruin your experiment and what to do about it. *American Journal of Political Science*, 62(3):760–775, 2018.

[15] C. N. Morris. Parametric empirical bayes inference: theory and applications. *Journal of the American statistical Association*, 78(381):47–55, 1983.

[16] National Governors Association Center for Best Practices, Council of Chief State School Officers. Common core state standards: Mathematics, 2010.

[17] J. F. Pane, B. A. Griffin, D. F. McCaffrey, and R. Karam. Effectiveness of cognitive tutor algebra i at scale. *Educational Evaluation and Policy Analysis*, 36(2):127–144, 2014.

[18] J. F. Pane, D. F. McCaffrey, M. E. Slaughter, J. L. Steele, and G. S. Ikemoto. An experiment to evaluate the efficacy of cognitive tutor geometry. *Journal of Research on Educational Effectiveness*, 3(3):254–281, 2010.

[19] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[20] G. Rasch. *Probabilistic models for some intelligence and attainment tests.* ERIC, 1993.

[21] S. W. Raudenbush and A. S. Bryk. *Hierarchical linear models: Applications and data analysis methods*, volume 1. sage, 2002.

[22] J. Roschelle, M. Feng, R. F. Murphy, and C. A. Mason. Online mathematics homework increases student achievement. *AERA open*, 2(4):2332858416673968, 2016.

[23] A. Sales, T. Patikorn, and N. T. Heffernan. Bayesian partial pooling to improve inference across a/b tests in edm. In *Proceeding of the Educational Data Mining Conference*, 2018.

[24] A. Sales, A. Wilks, and J. Pane. Student usage predicts treatment effect heterogeneity in the cognitive tutor algebra i program. In *Proceedings of the 9th International Conference on Educational Data Mining. International Educational Data Mining Society*, pages 207–214, 2016.

[25] W. J. van der Linden and R. K. Hambleton. *Handbook of modern item response theory*. Springer Science & Business Media, 2013.

APPENDIX

A. A SIMULATION STUDY OF MULTIPLE COMPARISONS

We ran a small simulation study testing [9]'s assertion that multiplicity corrections are unnecessary when estimating different effects from BLUPs in a multilevel model. [9] stated their case in terms of fully Bayesian models, whereas we used an empirical Bayesian approach that may differ somewhat.

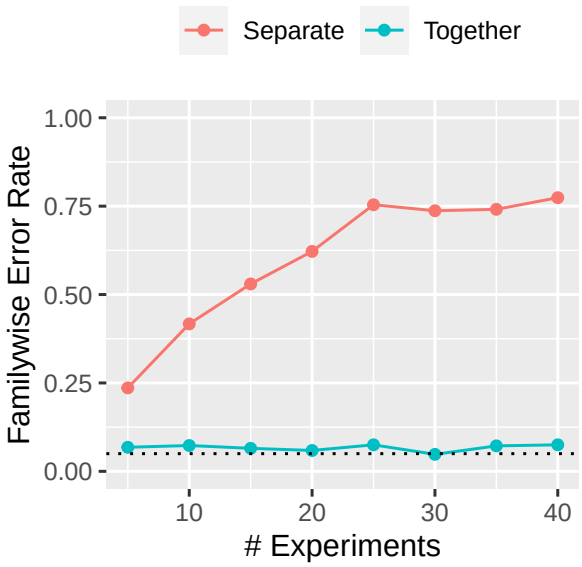


Figure 7: United we stand: results from a simulation of familywise error rate using separate t-tests for each experiment or using multilevel modeling.

In our simulation, in each simulation run, we generated data on $Nexpr$ experiments, where $Nexpr$ was a parameter we varied. In each experiment, there were $n = 500$ simulated subjects, half assigned to treatment and half to control. They were given "outcome" data $Y \sim N(0, 1)$, with no treatment effect.

We analyzed the experiment data in two ways. First, we estimated a p-value for each experiment separately, using t-tests. This is the conventional approach. Then, we we estimated a multilevel model:

$$Y_{ij} = \beta_0 + \gamma_{1j} Expr_j + \gamma_{2j} Trt_i + \epsilon_{ij}$$

where β_0 is an intercept, γ_{1j} are random intercepts for experiment, γ_{2j} is the treatment effect for experiment j, and ϵ_{ij} is a normally-distributed error term. $\boldsymbol{\gamma} \sim MVN\left(\{0, \gamma_{20}\}, \Sigma\right)$ where γ_{20} is the average effect across all experiments. The number of experiments in each simulation run, $Nexpr$, was varied from 5 to 40, in increments of 5. In each case, we estimated the familywise error rate, the probability of at least one statistically significant effect estimate (at $\alpha = 0.05$) across the $Nexpr$ experiments.

The results are in Figure 7. As expected, the familywise error rate increased rapidly when effects were estimated and tested separately in each of the $Nexpr$ experiments. When effects were estimated jointly in a multilevel model, in a way analogous to the method described in Section 3, the familywise error rate remained roughly constant as $Nexpr$ increased. However, the familywise error rate in the multilevel modeling approach was slightly elevated, ranging from roughly 0.05 to 0.075.

Math Operation Embeddings for Open-ended Solution Analysis and Feedback

Mengxue Zhang[1], Zichao Wang[2], Richard Baraniuk[2], Andrew Lan[1*]
[1]University of Massachusetts Amherst, [2]Rice University

ABSTRACT

Feedback on student answers and even during intermediate steps in their solutions to open-ended questions is an important element in math education. Such feedback can help students correct their errors and ultimately lead to improved learning outcomes. Most existing approaches for automated student solution analysis and feedback require manually constructing cognitive models and anticipating student errors for each question. This process requires significant human effort and does not scale to most questions used in homeworks and practices that do not come with this information. In this paper, we analyze students' step-by-step solution processes to equation solving questions in an attempt to scale up error diagnostics and feedback mechanisms developed for a small number of questions to a much larger number of questions. Leveraging a recent math expression encoding method, we represent each math operation applied in solution steps as a transition in the math embedding vector space. We use a dataset that contains student solution steps in the Cognitive Tutor system to learn implicit and explicit representations of math operations. We explore whether these representations can i) identify math operations a student intends to perform in each solution step, regardless of whether they did it correctly or not, and ii) select the appropriate feedback type for incorrect steps. Experimental results show that our learned math operation representations generalize well across different data distributions.

Keywords

Embeddings, Feedback, Math expressions, Math operations

1. INTRODUCTION

Math education is of crucial importance to a competitive future science, technology, engineering, and mathematics (STEM) workforce since math knowledge and skills are required in many STEM subjects [11]. One important way to help struggling students improve in math is to diagnose errors from student answers to math questions and deliver personalized support to help them correct these errors [1]. In short-answer questions, feedback of various types [39] can be deployed according to the specific incorrect final answers students submit, while in open-ended questions, feedback can be deployed at intermediate solution steps according to the specific actions they take and their outcomes [22]. In traditional educational settings, this feedback process relies on teachers going over student work, identifying errors, and providing feedback [15], which results in a labor-intensive process and a slow feedback cycle for students. Such a setting is even more limited as a result of the COVID-19 pandemic, which introduced new barriers to face-to-face interactions between teachers and students.

In intelligent tutoring systems, a more scalable approach to math feedback is to automatically deploy feedback based on students' final answers or certain incorrect intermediate solution steps. For example, in ASSISTments [12], teachers can create hints and feedback messages for specific incorrect student answers to short-answer questions that they anticipate [28], which the system can automatically deploy when students submit these incorrect answers. This crowdsourcing approach efficiently scales up teachers' effort so that they can benefit a large number of students without putting in additional effort. In many other systems such as Cognitive Tutor [34] and Algebra Notepad [27], researchers use cognitive models to anticipate student errors as results of buggy production rules or insufficient knowledge on key math concepts [20, 24]. They then develop corresponding feedback for intermediate solution steps in multi-step questions (e.g., those on equation solving). This cognitive model-based approach requires significant effort by domain experts and has shown to be highly effective in large-scale studies.

However, these approaches for student feedback are still limited in their generalizability to many math questions deployed in daily homeworks and practices. For the teacher crowdsourcing approach, hint and feedback messages have to be written for each individual question (or group of questions generated from the same template with different numerical values). For the cognitive model-based approach, a rigorous solution process has to be specified for each question with annotations on the math operations that should be applied at each solution step. However, questions used in many real-world educational settings do not come with such information; teachers simply adopt them from sources

*This work is supported by the National Science Foundation under grant IIS-1917713.

Mengxue Zhang, Zichao Wang, Richard Baraniuk and Andrew Lan "Math Operation Embeddings for Open-ended Solution Analysis and Feedback". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 216-227. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

such as textbooks and open education resources and assign them to students without developing corresponding feedback mechanisms. Moreover, past research has shown that a large portion of incorrect student answers cannot be anticipated by cognitive models [43], teachers/domain experts [8], or numerical simulations [37]. Therefore, it may be hard for high-quality feedback developed for questions used in intelligent tutoring systems to generalize to questions in the wild.

1.1 Contributions

In this paper, we develop *data-driven* methods that enable us to analyze step-by-step solutions to open-ended math questions. In contrast to existing methods that rely on a *top-down* approach, i.e., defining the structure of the solution process and anticipating student errors, we propose a *bottom-up approach*, i.e., using learned representations of *math expressions* and *math operations* to predict i) math operations in student solution steps and ii) the appropriate feedback for incorrect solution steps. We restrict ourselves to the specific domain of *equation solving* where the solution process consists of applying specific math operations between math expressions in consecutive steps; other subdomains of math such as algebra word problems [45] and questions involving graphs and geometry [16] are left as future work. Specifically, our contributions are:

- First, we characterize math operations by how they *transform* math expressions in the math embedding space in each solution step. We leverage recent work on learning *math symbol embeddings* from large-scale scientific formula data [46] to encode math expressions in student solutions: each math expression is mapped to a point in the *math embedding vector space*. We use synthetically generated data as well as solution steps generated by real students to learn the representation of each math operation. We explore several methods for learning both implicit and explicit math operation representations: a classification-based method that does not explicitly impose a structure on math operations, a linear model that assumes each operation is characterized by an *additive vector* in the embedding space, and a nonlinear model where math operations live in their own, interconnected embedding spaces.
- Second, we apply these math operation representation learning methods to a real-world student step-by-step solution dataset collected while student learn equation solving in an intelligent tutoring system, Cognitive Tutor [34]. We validate our math operation representation learning methods via two tasks: i) predicting the specific math operation the student intended to apply in a solution step from the math expressions before and after the step and ii) predicting the appropriate feedback deployed to students from the incorrect math expressions they enter. Quantitative results show that tree embedding-based math expression encoding methods outperform other encoding methods since they are able to explicitly capture the semantic and structural characteristics of math expressions. They also have better generalizability across different data distributions and remain effective across different question difficulty levels and even when student solutions steps contain errors.

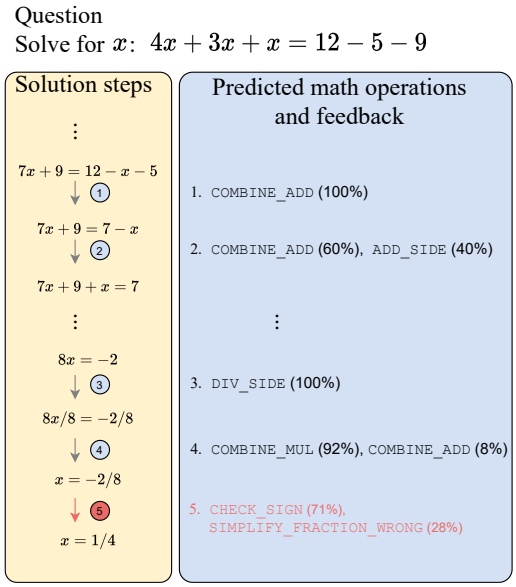


Figure 1. Demonstration of the generalizability of our math operation representations to other data sources for a solution process provided on Algebra.com. Our methods can successfully predict the math operations applied in each step and the appropriate feedback type in an incorrect step.

1.2 Use Case

Before diving into the technical details, we first illustrate a potential use case for our math operation representation learning methods and corresponding operation/feedback classifiers. Our goal is to transfer expert designs in intelligent tutoring systems for math education to questions in the wild. Specifically, we apply the math operation representations learned from student solution steps and corresponding labels (step name, feedback message) in the highly structured Cognitive Tutor system to environments that are not highly structured. Figure 1 shows the solution process to an equation solving question on Algebra.com[1] and the corresponding math operation and feedback predictions at each step. We see that our math operation representation learning methods can accurately predict the math operations applied in solution steps 1, 3, and 4 using the operation names provided in the Cognitive Tutor system. Even in step 2 where two different math operations are combined into a single step, i.e.,

$$
\begin{aligned}
&7x + 9 = 7 - x \\
&\downarrow \text{ADD } x \text{ TO BOTH SIDES} \\
&7x + 9 + x = 7 - x + x \\
&\downarrow \text{COMBINE TERMS ON RIGHT SIDE} \\
&7x + 9 + x = 7,
\end{aligned}
$$

despite only training on steps in Cognitive Tutor that involve only one math operation, the classifier is able to recognize both of them with high predictive probability for both. We

[1]The original question and the solution process can be found at `https://www.algebra.com/algebra/homework/equations/Equations.faq.question.4872.html`.

also change one of the solution steps, i.e., step 5, to make it incorrect and test our feedback classifier. In this case, the classifier is able to recognize the error in this step and find the corresponding feedback types in Cognitive Tutor. This potential use case demonstrates the utility of our math operation representation learning methods: by transferring knowledge learned in well-designed, highly-structured systems such as Cognitive Tutor, especially on what feedback to deploy for each student error, to other domains such as online math Q&A sites, we are *scaling up* the effort domain experts put into the design of these feedback mechanisms.

2. RELATED WORK

One related body of work in math education that studies student solution processes to identify student strategies and assess errors. Specifically, [33] uses inverse Bayesian planning to learn solution strategies (i.e., policies) in equation solving and capture student misunderstandings in a Markov decision process framework. Our work focuses on a different aspect of the solution process: the representation of the math expressions at each solution step and the modeling of the transitions between different math expressions under math operations. [9] uses basic math operations to construct programs to understand errors that students make in their solutions to arithmetic questions. Our work focuses on equation solving, which is a more difficult problem in which students responses are are more diverse and are less structured than arithmetic calculations.

Another related body of work focuses on learning representations of student answers to short-answer questions. [21] analyzes incorrect student answers across multiple questions, learn representations of errors, and generalize misconception feedback across questions. Our work analyze the full math expressions in intermediate solution steps while their work represents short answers according to the frequency they occur in an answer pool. [8] uses trained word embeddings to represent short answers for automated grading purposes. Our work focuses on learning transitions of math expressions across solution steps instead of learning representations of only the final answer.

In domains other than math education, there exist methods for automated feedback generation, including programming [30, 31, 40] and essays [35]. However, transferring these methods to math solutions is not trivial since i) open-ended math solutions are less structured than programming code and ii) data-driven representations of math symbols have not been developed until recently [46] whereas such representations have been studied for a long time in natural language processing [6, 7, 26].

Another body of remotely-related work focuses on using computer vision techniques to identify math expressions from images for similar math expression retrieval [29], turning hand-written math expressions into LaTeX [47], and automatically identifying and correcting student errors [14]. These works often bypass the inherent structure of math expressions and directly use an end-to-end model for their tasks, which means that they cannot be used to analyze student knowledge. Nevertheless, these techniques can be used to build large-scale datasets containing hand-written student solutions which we can use in the future.

3. BACKGROUND: EMBEDDING MATH EXPRESSIONS INTO VECTOR SPACES

In this section, we provide an overview of a recent method that we developed to embed math expressions into a vector space, i.e., a math embedding space. Doing so turns discrete, symbolic math expression representations into continuous, distributed representations [2], which enables us to manipulate math expressions in a manner compatible with modern machine learning methodologies.

Our embedding method is a *tree-structured* encoder illustrated in Figure 2. The key observation is that any math expression has a corresponding symbolic *tree-structured* representation in the *operator tree* format. In the operator tree, the non-terminal (non-leaf) nodes are math operators, i.e., addition and subtraction, and terminal (leaf) nodes are numbers or variables; See Figure 2 for an illustration. Thus, an operator tree explicitly captures the semantic and structural properties of a math expression. A number of existing works have demonstrated the superior performance of using operator tree representations of math expressions compared to other math expression representations in applications such as automatic math word problem solving [32, 48, 51] and math formulae retrieval [5, 25, 49, 50].

Therefore, we built a math expression encoder that leverages the operator tree representation of math expressions. Specifically, during the encoding process, it first converts a math expression into its corresponding tree format, using the parser introduced in [5]. It then linearizes the tree by depth first search that enables us to process nodes as a sequence in which each math symbol is associated with its own trainable embedding. Next, it leverages positional encoding, similar to [44, 38], to retain the relative position of each node in the tree. The output of our encoder is a fixed-dimensional embedding vector that represents the input math expression, which we will use to learn representations of math operations for the math operation classification and feedback prediction tasks. We pretrain the encoder on a large corpus of math expressions extracted from Wikipedia and arXiv articles and demonstrated superior performance in reconstructing math expressions (and scientific formulae) and retrieving similar expressions. See the anonymized version of our work at [46]. We will refer to the trained encoder as the *math expression encoding method* in what follows.

4. LEARNING REPRESENTATIONS OF MATH OPERATIONS

In this section, we detail methods we use to learn both implicit and explicit math operation representations by studying how they transform math expressions in each solution step in the math embedding space. In these methods, we leverage the math expression encoding method developed in our prior work that we reviewed above to embed math expressions into vectors and work with these embedding vectors. However, since these embeddings are trained on math expressions that are very different from those occurring in actual student solution steps, we use an additional trainable, fully-connected neural network to adapt these embeddings to our dataset, following a popular approach in natural language processing [13]. Specifically, we have $\mathbf{e} = g_{\boldsymbol{\gamma}}(\mathbf{m})$ where $\mathbf{m}$ and $\mathbf{e}$ are the embedded vector of a math expression

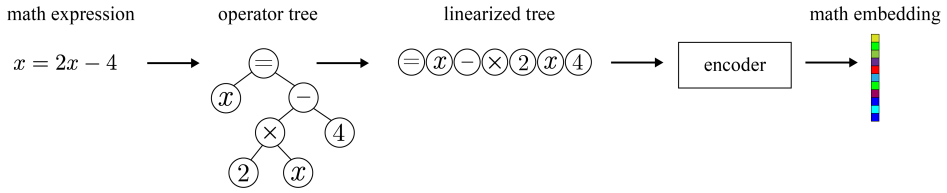


Figure 2. Illustration of the math expression encoding method that we employ in this work.

in our dataset before and after the adaptation, respectively. $\boldsymbol{\gamma}$ denotes the set of parameters in the fully-connected network that we will learn during the training process.

We define a step in a student's solution to open-ended math questions as a tuple $(\mathcal{E}_1, \mathcal{E}_2, z)$, where $z \in \mathbf{Z}$ is the math operation applied in this step, with $\mathcal{Z}$ denoting the set of possible math operations. $\mathcal{E}_1 \in \mathbf{E}$ and $\mathcal{E}_2 \in \mathbf{E}$ denote the math expressions involved in this step before and after applying this math operation, i.e., the step can be expressed as $\mathcal{E}_1 \xrightarrow{z} \mathcal{E}_2$. $\mathbf{E}$ denotes the set of all unique math expressions (across all steps in a dataset). For simplicity, we assume that only one math operation is applied in each step; an extension to cases where multiple math operations is trivial and will be discussed in what follows. $\mathbf{e}_1 \in \mathbb{R}^D$ and $\mathbf{e}_2 \in \mathbb{R}^D$ are the fine-tuned embedding vectors that correspond to math expressions $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively, where D is the dimension of the embedding.

4.1 Math Operation Classification

The first task we will study in this paper is to classify the math operation applied in a solution step given the math expression embeddings before and after appliying it, $\mathbf{e}_1$ and $\mathbf{e}_2$. The same notations and approaches also apply to our second task, feedback classification. This task can simply be solved using a supervised learning method, e.g., a regression model where the predicted probability of predicting a math operation $\hat{z}$ is given by

$$p(\hat{z} = z) = \text{softmax}(\mathbf{v}_z^T[\mathbf{e}_1^T,\ \mathbf{e}_2^T]^T),$$

where softmax$(\cdot)$ is the softmax function for multi-label classification [10]. $\mathbf{v}_z$ is a parameter vector associated with each math operation z, which is used to compute an inner product with the concatenation of $\mathbf{e}_1$ and $\mathbf{e}_2$ before being fed into the softmax function. On a training dataset with given tuples $(\mathbf{e}_1, \mathbf{e}_2, z)$, we can learn the parameters $(\mathbf{v}_z)$ by minimizing the cross-entropy loss [10] between the predicted math operation $\hat{z}$ and the actual math operation. This approach can be seen as learning *implicit* representations of math expressions since they are captured by the classifier parameters.

4.2 Learning Math Operation Representations

The classification approach we detailed above can help us classify the math operation applied in a solution step but falls short on learning *explicit* representations of math operations. The latter is important, however, to help us understand students' math solution processes and diagnose their errors. We now detail a series of methods for us to learn explicit representations of math operations.

4.2.1 Translating embeddings

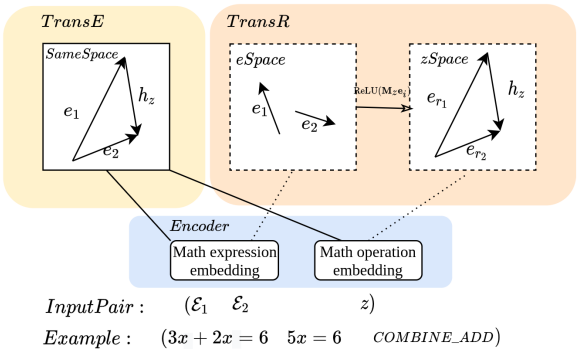


Figure 3. Illustration of the TransE and TransR frameworks. TransE puts the embeddings of equations e_1, e_2, and math operation z in the same embedding space, whereas TransR puts them in their own embedding spaces.

We will leverage the translating embedding (TransE) framework [3] that has found success in embedding entities and characterizing relationships between entities in multi-relational data. Our key assumption here in this framework is that math operations are *linear and additive*, i.e., the relationship between math expressions before and after a math expression satisfy

$$\mathbf{e}_2 \approx \mathbf{e}_1 + \mathbf{h}_z,$$

where $\mathbf{h}_z \in \mathbb{R}^D$ is the embedding of the *math operation* z. In other words, we assume that the effect of a math operation is characterized by the difference in the embedding vectors between the math expressions before and after it in a single step; adding it to the embedded vector of $\mathcal{E}_1$ results in the embedded vector of $\mathcal{E}_2$ after the step.

To learn these math operation embeddings from data, we use two loss functions. The first loss function promotes this linear and additive relationship between embeddings of the math expressions and operations on the training data. To this end, we define a distance function as $d(\mathbf{e}_1, \mathbf{e}_2, \mathbf{h}_z) = \|\mathbf{e}_1 + \mathbf{h}_z - \mathbf{e}_2\|_2^2$ and define the loss function as

$$L_1 = \sum_{(\mathcal{E}_1, \mathcal{E}_2, z)} d(\mathbf{e}_1, \mathbf{e}_2, \mathbf{h}_z).$$

The second loss function pushes counterfeit step tuples that are generated by replacing elements in an observed step tuple with other ones in the dataset to not satisfy the aforementioned linear and additive relationship. To this end, we minimize the pairwise marginal distance ranking-based loss

given by

$$L_2 = \sum_{(\mathcal{E}_1,\mathcal{E}_2,z)\in\mathbf{S}} \sum_{(\mathcal{E}_1',\mathcal{E}_2',z')\in\mathbf{S}'_{(\mathcal{E}_1,\mathcal{E}_2,z)}} [\gamma + d(\mathbf{e}_1,\mathbf{e}_2,\mathbf{h}_z) - d(\mathbf{e}_1',\mathbf{e}_2',\mathbf{h}_z')]_+,$$

where $[x]_+ = x$ when $x > 0$ and 0 otherwise and $\gamma > 0$ is a hyper-parameter that controls the margin of the distance ranking. $\mathbf{S}$ denotes the set of steps in the dataset and $\mathbf{S}'_{(\mathcal{E}_1,\mathcal{E}_2,z)}$ is a set of counterfeit steps that are perturbed versions of the actual step $(\mathcal{E}_1,\mathcal{E}_2,z)$, generated by randomly replacing one of the triplet elements in the step by a different math expression or math operation from another step, i.e.,

$$\begin{aligned}\mathbf{S}'_{(\mathcal{E}_1,\mathcal{E}_2,z)} &\sim A\cup B\cup C,\\ \text{where } A &= \{(\mathcal{E}_1',\mathcal{E}_2,z) : \mathcal{E}_1' \neq \mathcal{E}_1 \in \mathbf{E}\}\\ B &= \{(\mathcal{E}_1,\mathcal{E}_2',z) : \mathcal{E}_2' \neq \mathcal{E}_2 \in \mathbf{E}\}\\ C &= \{(\mathcal{E}_1,\mathcal{E}_2,z') : z' \neq z \in \mathbf{Z}\}.\end{aligned}$$

Intuitively speaking, our objective encourages the distance function calculated on an actual tuple in the dataset to be smaller than that calculated on a perturbed version of it. Figure 3 illustrates the whole process.

The final loss function that we minimize is simply the combination of these two loss functions as $L = L_1 + L_2$. Using the learned embeddings of each math operation, we can classify them from the math expressions $\mathcal{E}_1$ and $\mathcal{E}_2$ using the nearest neighbor classifier, i.e., $\hat{z} = \text{argmin}_z d(\mathbf{e}_1,\mathbf{e}_2,\mathbf{h}_z)$.

4.2.2 Learning Entity and Relation Embeddings

Despite potentially exhibiting excellent interpretability, TransE's assumption that math operations are linear and additive in the math expression embedding space may be too restrictive. This assumption puts math operations are vectors in the same latent space where similar math expressions will be close to each other. However, different math operations are fundamentally different and can transform the same math expression into dramatically different math expressions that are far apart in the embedding space. For example, different math operations can focus on transforming different parts of the same math expression. The steps $(3+5+2x = x+1,\ 8+2x = x+1$, combine similar terms) and $(3+5+2x = x+1,\ 3+5+2x-x = x+1-x$, subtract from each side) have the same starting math expression $\mathcal{E}_1$. In the first step, only similar terms on the left hand side of the equation are combined, regardless of the other side of the equation. In the second step, we subtracted x from both sides of the equation, which is a consequence of the equality symbol in the equation, which means that subtracting the same term on both sides of the equation but not what exactly is on each side. Therefore, TransE's linear and additive assumption means that the resulting $\mathcal{E}_2$ in these steps will be very different due to the different math operations applied, which conflicts with the observation that they are very similar. To address this limitation, we explore the Learning Entity and Relation Embeddings (TransR) [23] model, which models math expressions and math operations in different spaces, i.e., there will be a shared embedding space for all math expressions but separate relation spaces for different math operations.

TransR learns the embeddings of math operations by projecting them to their corresponding relation spaces and then learning translations between those projected expressions. For each math operation z, we set a projection matrix $\mathbf{M}_z \in \mathbb{R}^{D\times D}$ that projects a math expression to its relation space. To make this projection nonlinear, we apply the rectified linear unit (ReLU) activation function [10] to it and define the corresponding distance function as

$$d_z(\mathbf{e}_1,\mathbf{e}_2,\mathbf{h}_z) = \|\text{ReLU}(\mathbf{M}_z\mathbf{e}_1) + \mathbf{h}_z - \text{ReLU}(\mathbf{M}_z\mathbf{e}_2)\|_2^2.$$

Correspondingly, the two loss functions in the TransR framework are given by

$$\begin{aligned}L_1 &= \sum_{(\mathcal{E}_1,\mathcal{E}_2,z)} d_z(\mathbf{e}_1,\mathbf{e}_2,\mathbf{h}_z),\\ L_2 &= \sum_{(\mathcal{E}_1,\mathcal{E}_2,z)\in\mathbf{S}} \sum_{(\mathcal{E}_1',\mathcal{E}_2',z')\in\mathbf{S}'_{(\mathcal{E}_1,\mathcal{E}_2,z)}} [\gamma + d_z(\mathbf{e}_1,\mathbf{e}_2,\mathbf{h}_z) - d_z(\mathbf{e}_1',\mathbf{e}_2',\mathbf{h}_z')]_+.\end{aligned}$$

The projection matrices $\mathbf{M}_z$, $\forall z \in \mathbf{Z}$ are included as part of the trainable parameters. The rest of the training and resulting math operation classification procedure remains unchanged from the TransE framework.

5. EXPERIMENTS

We now detail a series of quantitative and qualitative experiments that we have conducted to validate the learned representations of math operations. Using the Cognitive Tutor 2010 equation solving (CogTutor) dataset,[2] we focus on two tasks: i) classifying the math operation a student applies in a solution step and ii) classifying the feedback category corresponding to certain types of incorrect steps, from the math expressions the student enters before and after the step.

5.1 Dataset

We use the CogTutor dataset which we accessed via the PSLC DataShop [19]. The dataset contains detailed tutor logs generated as students in a school use the Cognitive Tutor system [34] for their Algebra I class. These logs contain the students' step-by-step solutions to equation solving problems, where each step is a tuple with three elements: a *math expression* $\mathcal{E}_1$ at the beginning of the step, the *step name* z, i.e., the math operation the student selected to apply to this math expression, and the resulting math expression $\mathcal{E}_2$ after the step. Students can select math operations from a built-in list in Cognitive Tutor: COMBIN_ADD, COMBINE_MUL, ADD_SIDE, SUB_SIDE, MUL_SIDE, DIV_SIDE, and DISTRIBUTE; see Table 1 for an illustration of these operations and some examples of the corresponding math operations before and after them in a step.

There are a total of 50,406 steps in this dataset that can be further divided into three subsets according to their outcomes: `OK` (43,413 steps), `ERROR` (6,377 steps), and `BUG` (5,744 steps). The `OK` subset contains steps that are correct, i.e., the student both selected the correct math operation and arrived at the correct math expression. The `BUG` and `ERROR` subsets contain incorrect student steps, either because the operation they selected was incorrect or

[2] `https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=660`

Step (Math operation)	Description	Example
COMBINE_ADD	combine two similar terms with add/sub operator	$3x + 2x \rightarrow 5x$
COMBINE_MUL	combine two similar terms with multiply/divide operator	$x * x \rightarrow x^2$
ADD_SIDE	add a math term on each side	$x = 1 \rightarrow x + 1 = 1 + 1$
SUB_SIDE	subtract a math term on each side	$x = 1 \rightarrow x - 1 = 1 - 1$
MUL_SIDE	multiply a math term on each side	$x = 1 \rightarrow x * 2 = 2$
DIV_SIDE	divide a math term on each side	$x = 1 \rightarrow x/2 = 1/2$
DISTRIBUTE	distribute(expand) the terms	$(x + 1)x \rightarrow x * x + x$

Table 1. Detailed descriptions and examples for each math operation in the CogTutor dataset.

because they selected the correct operation but did not apply it correctly, i.e., arriving at an incorrect math operation after the step. The difference between these two subsets is that `BUG` contains steps that fit one of the predefined error templates in the Cognitive Tutor system; in this case, the system can automatically diagnose the error and deploy a predefined feedback. On the other hand, `ERROR` contains incorrect steps that Cognitive Tutor could not automatically diagnose the underlying error. The `OK` subset can be further split into six predefined difficulty levels (named as ES_01,ES_02, ES_03 ,ES_04, ES_05, and ES_07), with 2,068, 7,546, 8,183, 13,393, 5,484, and 2,801 steps, respectively. We do not further split the `BUG` and `ERROR` subsets for the math operation classification task due to their limited sizes.

To learn the representation of math operations, we need examples of how they transform one math expression into another. However, the CogTutor dataset may not contain enough data that is rich in both quantity and diversity for neural network-based models to learn from. Therefore, we designed a synthetic data generator stemming from the math question answering dataset created by DeepMind [36]. The generator can generate steps by first generating the initial math expression and then applying math operations listed in Table 1 to arrive at a resulting math expression. We have full control over the generated steps through the entropy, degree, and flip parameters. Increasing entropy introduces more complexity to the math expressions as numerical constants generated get larger. Increasing the degree parameter introduces monomials of higher degrees and also adds more terms in the math expression. Finally, the flip parameter allows us to control which side of an equation has a higher chance to be more complicated than the other. Tuning these parameters within this flexible synthetic data generation method enables us to generate a large amount of steps that closely resembles those in the CogTutor dataset.

5.2 Methods

To fully evaluate the effectiveness of our math operation representations, we also experiment with two other ways of encoding math expressions commonly used in natural language processing tasks, in addition to the tree embedding-based and translation-based encoder that we introduced in Section 4.2. These two encoders include a gated recurrent unit (GRU)-based encoder [4] and a convolutional neural network (CNN)-based encoder [17]; we will use the output of these encoders to replace $[\mathbf{e}_1^T,\ \mathbf{e}_2^T]^T$ as input to the classifier detailed in Section 4.1.

Specifically, these two encoders first concatenates the two math expressions before and after the step, i.e., $\mathcal{E} = [\mathcal{E}_1, \mathcal{E}_2]$. For each character x_t in $\mathcal{E}$, we compute its embedding

$$\boldsymbol{x}_t = \mathbf{W}^T \text{onehot}(x_t)\,,$$

where $\mathbf{W}$ is a trainable embedding matrix. Using these character embeddings, the GRU encoder computes

$$\boldsymbol{h}_t = \text{GRU}_\theta(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})\,,$$

where θ represents all the trainable parameters in GRU. We then replace $[\mathbf{e}_1^T,\ \mathbf{e}_2^T]^T$ with $\boldsymbol{h}_T$ as input to the classifier where T is the total number of characters in $\mathcal{E}$. Similarly, the CNN encoder computes

$$\boldsymbol{h} = \text{max\_pool}(\text{CNN}_\phi([\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T]))\,,$$

where CNN_ϕ represents a 2D CNN with parameters ϕ and max_pool is a 1D max pooling operator. Combined, they return a fixed dimensional feature vector $\boldsymbol{h}$ that replaces $[\mathbf{e}_1^T,\ \mathbf{e}_2^T]^T$ as input to the classifier. For each of these two models, we learn its parameters jointly with the classification task using the cross-entropy loss that we described in Section 4.1.

Overall, we test five different methods for the math operation classification and feedback classification tasks. The first three methods use different encoding methods in conjunction with a classifier: i) using the GRU encoder to encode math expressions as input to the classifier, which we dub **GRU+C**, ii) using the tree embedding-based encoder instead, which we dub **TE+C**, and iii) using the CNN encoder instead, which we dub **CNN+C**. These methods do not learn explicit representations of math operations. The next two methods use the TransE and TransR frameworks to learn these representations using tree embeddings: iv) using tree embedding-based encoder as input to the TransE framework in conjunction with a nearest neighbor classifier, which we dub **TE+TransE**, and v) using the TransR framework instead of the TransE framework to study math operations in multiple relation spaces, which we dub **TE+TransR**.

5.3 Experimental Setup

We first test our math operation representation learning methods on the `OK` subset via 5-fold cross-validation, i.e., training on 80% of steps in the subset to learn representations of math operations and testing them on the remaining 20%. We also test the generalizability of the learned representations to incorrect steps, i.e., replace the test set with the `ERROR` and `BUG` subsets, and check whether we can still recognize the math operation a student applied in an incorrect step. The results are detailed in Section 5.4.1.

Since the distribution of math expressions in the `OK`, `ERROR` and `BUG` data subsets are mostly similar with minor differ-

	OK	ERROR	BUG
GRU+C	99.18 ± 0.23	93.87 ± 0.66	95.89 ± 0.63
TE+C	99.82 ± 0.04	93.30 ± 0.65	95.38 ± 0.62
CNN+C	95.37 ± 0.44	86.82 ± 1.38	91.02 ± 0.59
TE+TransE	96.27 ± 0.17	86.32 ± 1.23	84.21 ± 2.13
TE+TransR	99.17 ± 0.21	91.28 ± 1.12	91.31 ± 1.87

Table 2. Math operation classification accuracy for all methods training on the OK subset of the CogTutor dataset and testing on different data subsets. Accuracy is high across the board, while GRU-based encoding and tree embedding-based encoding in conjunction with a classifier result in the best performance.

ences, the previous experiment does not give us a good idea on the generalization ability of our math operation representation learning methods. Therefore, we further divide the OK subset into six smaller subsets, each corresponds to a different difficulty level (with different structure and complexity) according to questions within it, and test the generalizability of the learned math operation representations. The results are detailed in Section 5.4.2. In practice, solution step data generated by real students is often limited. Therefore, we conduct two more experiments to test whether synthetically generated steps can help us learn math operation representations that generalize to real data. First, we repeat the experiments above using synthetically generated steps as the training set. This synthetic training set consists of $1,000$ steps for each math operation defined in Table 1 (adding up to a total of $7,000$ across different difficulty levels). The results are detailed in Section 5.4.3. Second, to study the impact of synthetically generated data when real data is limited, we pre-train the math operation representations with synthetic data, fine-tune on a small amount of real data from each difficulty level in the OK subset, and test on the rest. The results are detailed in Section 5.4.4.

To test the ability of our learned math operation representations on recognizing student errors, we use them to classify feedback types provided by CogTutor in the BUG data subset. Examples of such errors include when a student calculated the wrong simplification result, used the wrong sign in front of terms, and applied useless/unlogical steps to solve the problem, etc. The results are detailed in Section 5.4.5.

We use Adam optimizer [18] with learning rate 0.001, batch size 64 and run 10 training epochs for each experiment. The math expression encoder outputs length-512 embedding vectors for each math expression, which we adapt to length-32 embedding vectors dimensions using a trainable fully-connected neural network. All of our experiments were conducted on a server with a single Nvidia RTX8000 GPU.

5.4 Results and Discussion

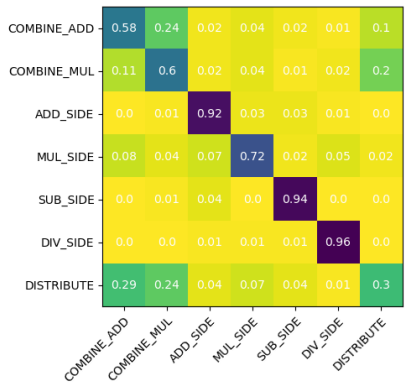
5.4.1 Generalizing to incorrect steps

Table 2 shows the averages and standard deviations of math operation classification accuracy for every method we experimented with using the OK subset as the training set. As expected, testing on the ERROR and BUG subsets result in slightly lower (5-10%) math operation classification accuracy for all methods since the training set does not contain incorrect steps. However, even on steps that are incorrect, these methods can still effectively identify the math operation a student intended to apply (with up to 95% accuracy), suggesting that they may be applicable to fully open-ended question solving solutions that are not highly structured, unlike those in Cognitive Tutor, to provide feedback to teachers on students' solution approaches.

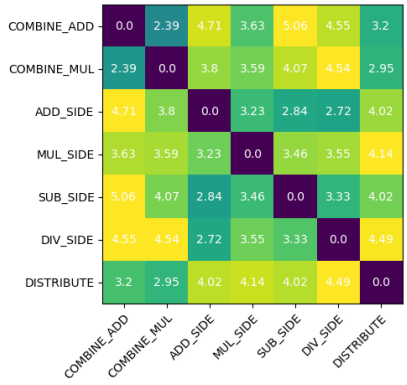
We observe that using GRUs and tree embeddings as representations for math expressions and applying a classification method on top of these representations result in similar performances; GRUs slightly outperform tree embeddings in cases where we use the ERROR and BUG subsets as the test set while tree embeddings slightly outperform GRUs in the case where we use a part of the OK subset as the test set. Using CNNs to encode math expressions as input to a classifier results in worse performance, suggesting that they do not capture the semantic and structural information in math expressions as well as GRUs and tree embeddings. As expected, using tree embeddings under the TransE and TransR frameworks leads to worse performance than the first two methods, with TransE achieving low performance (especially on the BUG subset) and TransR achieving comparable performance to the classification-based methods on the OK subset but lower performance on the ERROR and BUG subsets. This result can be explained by the additional structural restriction that math operations are represented as linear and additive in some embedding space in the TransE framework, which makes it less robust against incorrect student solution steps. Using the TransR framework mitigates this problem due to its use of different relation spaces for each math operation.

These methods perform similarly in the math operation classification task on real data largely due to the limited variation and complexity in the math expressions. The Cognitive Tutor system limits the degrees of freedom in a students' response by splitting an open-ended step into the separate actions of selecting a single math operation and entering the resulting math expression, which limits the variability in the data. In the next experiment, we see that when we control against different levels of complexity in these math expressions and forcing these methods to generalize across complexities, their performance vary significantly.

Figure 4 visualizes the confusion matrix for math operation classification on the OK subset and the pairwise euclidean distances between math operation embeddings learned via the TransE framework using tree embeddings for math expressions. Rows correspond to the true math operations applied in steps and columns correspond to predicted ones. Percentages in the confusion matrix (Figure 4a) are normalized w.r.t. the number of appearances of each math operation. We see that our math operation representation learning method captures some meaning of these operations (Figure 4b); the learned math operation embeddings capture the structural changes in math expression in ways that match our intuition. For instance, both COMBINE_ADD and COMBINE_MUL can be considered types of simplifications, so the Euclidean distance between the learned embeddings for these two operations is low. This observation is not surprising due to the similar nature of these operations. Moreover, COMBINE_ADD, COM-



(a) Confusion matrix of math operations classification.



(b) Euclidean distance between learned math operation embedding vectors.

Figure 4. Details of TE+TransE for the math operation classification task on the OK subset. These results match our intuition on how these math operations are related.

BINE_MUL, and DISTRIBUTE are often confused with one another. These results are also validated by a 2-D visualization (using t-SNE [42] as a dimensionality reduction method) of the learned math operation embeddings in Figure 5, where different math operations are mostly well separated except for COMBINE_ADD, COMBINE_MUL, and DISTRIBUTE. One possible explanation is that these operations are all applied to one side of the equation during a solution step, leaving one side of the equation unchanged, while the other operations, such as ADD_SIDE, SUB_SIDE, MUL_SIDE, and DIV_SIDE are all applied to both sides of the equation. Therefore, this result suggests that tree embeddings enable us to characterize a math operation by the *structural* change in math expressions before and after a solution step where it is applied. Furthermore, the classification accuracy for the DISTRIBUTE operation is significantly lower than that for other operations. This result is likely due to the fact that the number of steps with this operation is significantly lower than that for other operations.

5.4.2 *Generalizing to different difficulty levels*

In this experiment, we test the ability of our learned math operation representations to generalize to math expressions with different levels of complexity in questions at different levels of difficulty. Although they are all about equation solving, questions at different difficulty levels in Cognitive Tutor involve math expressions that look very different. For example, in the easiest level (ES_01), the equation that needs to be solved in a question looks like $x + 5 = 9$, with only a single variable and without numbers with decimals. In contrast, in the hardest level (ES_07), a question may contain coefficients with several decimal places and multiple variables, such as solve for m in the equation $m(k - n) = gs$. We only compare the GRU-based encoder and the tree embedding-based encoder in conjunction with a classifier since they are the best performing methods in the previous experiment. Table 3 lists the math operation classification accuracy for both methods after training on steps at different difficulty levels in the OK subset and testing on steps at other difficulty levels (including incorrect ones). We see that TE+C overall outperforms GRU+C in almost every case. This results suggest that tree embeddings are effective at capturing the structural property of a math expression. As a result, math operation representations based on tree embeddings excel at capturing the structural *change* in math expressions before and after applying a math operation, leading to better generalizability than GRU-based encoding that do not explicitly account for this change.

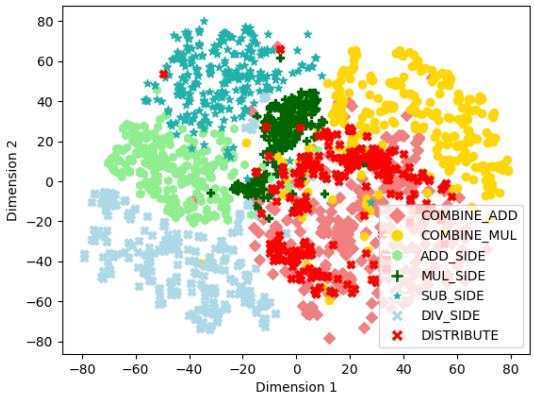


Figure 5. Visualization of learned math expression change for a randomly sampled subset of student solution steps in 2-D and corresponding operations (best viewed in color).

5.4.3 *Generalizing to different data distributions*

In this experiment, we test the ability of our methods to generalize from synthetically generated data to real student data. We train different math operation classification methods on the 2,000 synthetically generated steps and test them on steps generated by real students in the CogTutor dataset. Table 4 shows the mean and standard deviation for each method on each real data subset. We see that TE+C significantly outperforms GRU+C and CNN+C on all data subsets, which is in stark contrast to the previous experiment where the difference in performance across all methods is much smaller. This observation suggests that tree embeddings are more effective at capturing the semantic/structural effect of math operations on math expressions, thus generalizing better to different data distributions. Indeed, although the synthetically generated steps and the real steps have the same set of math operations, the distributions of numbers

Train on	Method	OK	ERROR	BUG
ES_01	GRU+C	58.82 ± 1.12	63.74 ± 1.13	66.02 ± 1.12
	TE+C	76.51 ± 0.62	84.24 ± 0.87	67.49 ± 1.10
ES_02	GRU+C	71.05 ± 1.12	76.66 ± 1.11	69.01 ± 1.14
	TE+C	87.89 ± 0.34	93.96 ± 0.72	80.44 ± 0.78
ES_03	GRU+C	82.39 ± 3.93	79.24 ± 1.47	80.01 ± 1.67
	TE+C	90.79 ± 1.12	93.83 ± 1.32	84.70 ± 1.54
ES_04	GRU+C	76.72 ± 0.14	71.35 ± 6.12	83.32 ± 2.24
	TE+C	94.65 ± 0.12	92.72 ± 1.32	90.99 ± 1.72
ES_05	GRU+C	81.74 ± 0.33	73.36 ± 1.69	78.36 ± 1.07
	TE+C	87.66 ± 0.25	80.00 ± 1.32	77.81 ± 0.99
ES_07	GRU+C	76.25 ± 3.21	73.15 ± 3.42	67.35 ± 3.62
	TE+C	79.44 ± 0.62	79.29 ± 0.72	72.53 ± 2.26

Table 3. Math operation classification accuracy after training on steps with different difficulty levels and testing on the OK ERROR, and BUG subsets. Tree embedding-based encoding outperforms GRU-based encoding.

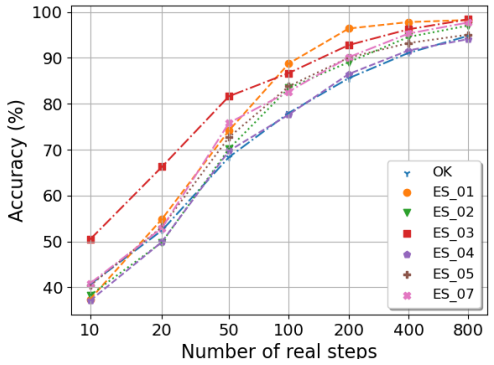


Figure 6. Math operation classification accuracy for the TE+C method when real data is limited. Using synthetically generated steps as a starting point, we already start with acceptable classification accuracy even with few real steps generated by students. The performance steadily improves after more real data becomes available.

$(1, 0.5, -7,$ etc.) and variables $(x, u, t,$ etc.), resulting in a mismatch between the data distributions. Tree embedding-based methods benefit from the tree-based representations of math expressions that can effectively capture structural information, making it easy for the learned embeddings of math expressions to generalize to unseen data.

5.4.4 *Generalizing from synthetic data*

Ideally, if there is a large amount of training data, i.e., steps generated by real students containing different types of math expressions and detailed labels on these steps such as the math operation(s) applied, the error(s) if a step is incorrect, and corresponding feedback, we can simply use that data to learn our math operation representations. However, in practice, the amount of real data is often limited. Figure 6 plots the performance of TE+C on all subsets of the CogTutor dataset, training on a portion of steps in the subset for training and testing on the rest. We see that the performance on math operation classification suffers considerably when we only have limited training data. Therefore, synthetically generated data can play a vital role in improving their performance under this circumstance; the strategy of fine-tuning models trained on synthetically generated data using a small amount of real data can be effective. Specifically, we start with a pre-trained math operation classification model on the 7000 synthetically generated steps and fine tune it on a small number of real steps by doing gradient descent on these steps for 10 epochs. Figure 7 plots the improvement in math operation classification accuracy for the fine-tuned model over the model that trains on only real data of various amounts on all data subsets. We see that the pre-trained models always performs better, with significant improvement when the real data is extremely limited. This result suggests that i) effectively leveraging synthetically generated data can mitigate the problem of limited real data and ii) our math operation representation learning methods are capable of generalizing across different data distributions (synthetic $\rightarrow$ real).

	OK	ERROR	BUG
GRU+C	62.89 ± 3.93	64.06 ± 4.70	62.94 ± 2.24
TE+C	83.79 ± 0.14	75.49 ± 0.90	75.16 ± 0.55
CNN-C	51.12 ± 1.64	45.52 ± 0.98	59.82 ± 1.68
TE + TransE	80.17 ± 2.32	71.86 ± 3.24	72.32 ± 2.72
TE + TransR	82.22 ± 2.88	73.83 ± 3.46	74.85 ± 3.23

Table 4. Math operation classification accuracy for all methods training on 7,000 synthetically generated steps and testing on different subsets of the CogTutor dataset. Tree embedding-based methods significantly outperform other methods, showing better ability to generalize to different data distributions.

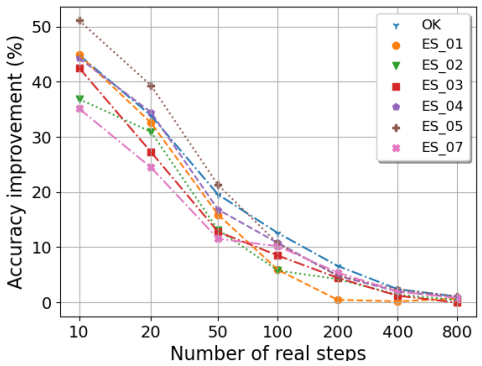


Figure 7. Math classification accuracy (difference in percentage) for TE+C, pre-training on synthetic data before fine-tuning on real data versus training only on real data. When real data is limited, pre-training on synthetic data results in significantly better performance.

5.4.5 *Feedback type classification*

In this experiment, we evaluate our math operation representation learning methods on the feedback type classification task. These feedback items were automatically deployed by Cognitive Tutor for incorrect steps in the BUG subset. We pre-processed these steps and grouped the detailed feedback items according to the students' errors that each

Method	Accuracy
GRU+C	75.35 ± 1.41
TE + C	78.71 ± 1.74
CNN+C	67.23 ± 1.54
TE + TransE	69.15 ± 1.13
TE + TransR	73.21 ± 1.63

Table 5. Feedback type classification accuracy for all methods on the BUG subset. Tree embedding-based encoding outperforms other encoding methods while TransE and TransR frameworks do not reach similar performance levels due to shortage of training data.

feedback item addresses and narrowed it down to a total of 24 types that occur multiple times. We perform 5-fold cross validation on this subset. Table 5 shows the averages and standard deviations of feedback classification accuracy for all methods on this task across the five folds. We see that due to the limited size of the BUG subset (only $5,744$ steps) and the high number of classes (24), all method perform worse than they do on the math operation classification task. Specifically, we see that the tree embedding-based encoder in conjunction with a classifier performs best while GRU-based encoding also performs well. This result shows that although tree embeddings are superior at capturing the meaning of math expressions, their advantage over simple encoding methods such as GRU-based encoding decreases due to increased noise in the data; some math expressions submitted by students in incorrect steps are ill-posed and do not make sense. Using the TransE and TransR frameworks result in slightly worse performance than classifiers since these methods explicitly learn a representation for each math operation, which limits their performance on this task due to the shortage of training data. However, since they capture the structural difference in math expressions before and after the step, they can cancel out some of the noise in erroneous steps, resulting in acceptable performance.

5.5 Discussions

Overall, we find that the GRU-based and tree embedding-based math expression encoders in conjunction with a classifier perform almost equally well in most situations, while the CNN-based encoder performs worse. The tree embedding-based encoder has stronger generalizability across different data distributions. We believe that as the math expressions and operations get more complicated, methods that leverage the tree structure of math expressions would be more advantageous. We also observe that TransR outperforms TransE most of the time, although in some experiments using TransE and TransR to explicitly learn math operation embeddings lead to slightly worse performance than classifiers using implicit representations of math expressions. However, TransE and TransR are much more powerful and enable us to study more tasks such as clustering solution steps and identifying typical student errors and learning solution *strategies*; See Section 6 for a detailed discussion.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we developed a series of methods to learn representations of math operations by observing how math expressions change as a result of these operations in step-by-step solutions to open-ended math questions. Our methods leverage math expression encoding methods that map tree-structured math expressions into a math embedding vector space. We demonstrated the effectiveness of our methods on a dataset containing detailed student solution steps to equation solving questions in the Cognitive Tutor system on two tasks: i) classifying the math operation applied in each step and ii) classifying the feedback the system deploys for each incorrect step. Results show that our learned math operation representations are meaningful and can often effectively generalize across different data distributions such as questions with different difficulty levels.

However, the success of our methods heavily depends on the availability of diverse large-scale training data. The Cognitive Tutor dataset that we used in this work represents a heavily restricted solution process since the list of math operations a student can apply in a step is pre-defined. Therefore, additional work has to be done to extend our method to truly open-ended step-by-step solution processes that are less structured. Moreover, our methods are restricted to a single solution step only and do not consider the relationship across multiple steps, which is related to another important aspect of solving open-ended math questions: the overall solution strategy, i.e., which math operation to apply next. Furthermore, in both classification tasks, using tree embeddings to encode math expressions in conjunction with a classifier outperforms explicitly learning vectorized representations of math operations in the TransE and TransR frameworks. However, these explicit representations may enable us to perform other tasks such as Nevertheless, our work provides a series of tools to analyze the math expressions students write down in their solutions by bridging the gap between symbolic math representations with continuous representations in vector spaces, enabling the use of state-of-the-art neural network-based methods. We believe that this work can potentially open up a new line of research that studies how to automatically analyze student solutions for grading and feedback purposes.

There are many avenues of future work. First, since most real-world open-ended solutions contain a mixture of math expressions and text, there is a need to learn a joint representation of math expressions and text in a shared embedding space. Second, this joint representation will enable us to train automated feedback generation methods in an end-to-end manner, using sequence-to-sequence learning methods [41]. Third, using learned math expression representations as the states and learned math operation representations from the TransE and TransR frameworks as the state transition model, we can apply reinforcement learning and inverse reinforcement learning methods to learn solution strategies, i.e., which math operation to apply in the next step. We can also study solution strategies employed by real students [33] and diagnose their errors and design corresponding feedback mechanisms to improve their learning outcomes. These future work directions will enable us to tap into the full potential of explicit math operation representations, which is not fully demonstrated in this paper: on the CogTutor dataset, the only relevant real-world dataset we found, we could only evaluate these explicit representations on the math operation and feedback prediction tasks, where they may not outperform tree embedding-based classification-based methods.

7. REFERENCES

[1] D. M. Adams, B. M. McLaren, K. Durkin, R. E. Mayer, B. Rittle-Johnson, S. Isotani, and M. Van Velsen. Using erroneous examples to improve mathematics learning with a web-based tutoring system. *Computers in Human Behavior*, 36:401–411, 2014.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

[3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[4] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proc. Conf. Empirical Methods Natural Language Process.*, pages 1724–1734, October 2014.

[5] K. Davila and R. Zanibbi. Layout and semantics: Combining representations for mathematical formula search. In *Prof. Intl. ACM SIGIR Conf. Res. Develop. Info. Retrieval*, page 1165–1168, 2017.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[7] S. T. Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

[8] J. A. Erickson, A. F. Botelho, S. McAteer, A. Varatharaj, and N. T. Heffernan. The automated grading of student open responses in mathematics. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 615–624, 2020.

[9] M. Q. Feldman, J. Y. Cho, M. Ong, S. Gulwani, Z. Popović, and E. Andersen. Automatic diagnosis of students' misconceptions in k-8 mathematics. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[11] S. Grover and R. Pea. Computational thinking in k–12: A review of the state of the field. *Educational researcher*, 42(1):38–43, 2013.

[12] N. T. Heffernan and C. L. Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4):470–497, 2014.

[13] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[14] Y. Hu, Y. Zheng, H. Liu, D. Jiang, Y. Liu, and B. Ren. Accurate structured-text spotting for arithmetical exercise correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 686–693, 2020.

[15] K. Kelly, N. Heffernan, S. D'Mello, J. Namais, and A. Strain. Adding teacher-created motivational video to an its. In *Proceedings of 26th Florida Artificial Intelligence Research Society Conference*, pages 503–508. Citeseer, 2013.

[16] A. Kembhavi, M. Seo, D. Schwenk, J. Choi, A. Farhadi, and H. Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition*, pages 4999–5007, 2017.

[17] Y. Kim. Convolutional neural networks for sentence classification. In *Proc. Conf. Empirical Methods Natural Language Process.)*, pages 1746–1751, Oct. 2014.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. on Learn. Representations*, 2015.

[19] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the edm community: The pslc datashop. *Handbook of educational data mining*, 43:43–56, 2010.

[20] K. R. Koedinger, A. Corbett, et al. *Cognitive tutors: Technology bringing learning sciences to the classroom*. na, 2006.

[21] J. Kolb, S. Farrar, and Z. A. Pardos. Generalizing expert misconception diagnoses through common wrong answer embedding. *International Educational Data Mining Society*, 2019.

[22] A. S. Lan, D. Vats, A. E. Waters, and R. G. Baraniuk. Mathematical language processing: Automatic grading and feedback for open response mathematical questions. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*, pages 167–176, 2015.

[23] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[24] R. Liu, R. Patel, and K. R. Koedinger. Modeling common misconceptions in learning process data. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, pages 369–377, 2016.

[25] B. Mansouri, S. Rohatgi, D. W. Oard, J. Wu, C. L. Giles, and R. Zanibbi. Tangent-cft: An embedding model for mathematical formulas. In *Proc. Intl. ACM SIGIR Conf. Res. Develop. Info. Retrieval*, page 11–18, 2019.

[26] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proc. Conference on Advances in Neural Information Processing Systems*, pages 3111–3119, Dec. 2013.

[27] E. O'Rourke, E. Butler, A. D. Tolentino, and Z. Popović. Automatic generation of problems and explanations for an intelligent algebra tutor. In *International Conference on Artificial Intelligence in Education*, pages 383–395. Springer, 2019.

[28] T. Patikorn and N. T. Heffernan. Effectiveness of

crowd-sourcing on-demand assistance from teachers in online learning platforms. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pages 115–124, 2020.

[29] L. Pfahler, J. Schill, and K. Morik. The search for equations–learning to identify similarities between mathematical expressions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 704–718. Springer, 2019.

[30] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas. Learning program embeddings to propagate feedback on student code. In *International conference on machine Learning*, pages 1093–1102. PMLR, 2015.

[31] T. W. Price, Y. Dong, and T. Barnes. Generating data-driven hints for open-ended programming. *International Educational Data Mining Society*, 2016.

[32] J. Qin, L. Lin, X. Liang, R. Zhang, and L. Lin. Semantically-aligned universal tree-structured solver for math word problems. In *Proc. Conf. Empirical Methods Natural Lang. Process.*, pages 3780–3789, Nov. 2020.

[33] A. N. Rafferty, R. A. Jansen, and T. L. Griffiths. Assessing mathematics misunderstandings via bayesian inverse planning. *Cognitive science*, 44(10):e12900, 2020.

[34] S. Ritter, J. R. Anderson, K. R. Koedinger, and A. Corbett. Cognitive tutor: Applied research in mathematics education. *Psychonomic bulletin & review*, 14(2):249–255, 2007.

[35] R. D. Roscoe, E. L. Snow, L. K. Allen, and D. S. McNamara. Automated detection of essay revising patterns: Applications for intelligent feedback in a writing tutor. *Grantee Submission*, 10(1):59–79, 2015.

[36] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli. Analysing mathematical reasoning abilities of neural models, 2019.

[37] D. Selent and N. Heffernan. Reducing student hint use by creating buggy messages from machine learned incorrect processes. In *International conference on intelligent tutoring systems*, pages 674–675. Springer, 2014.

[38] V. Shiv and C. Quirk. Novel positional encodings to enable tree-based transformers. In *Proc. Intl. Conf. Neural Info. Process. Syst.*, pages 12081–12091, 2019.

[39] V. J. Shute. Focus on formative feedback. *Review of educational research*, 78(1):153–189, 2008.

[40] R. Singh, S. Gulwani, and A. Solar-Lezama. Automated feedback generation for introductory programming assignments. In *Proc. 34th ACM SIGPLAN Conf. on Programming Language Design and Implementation*, volume 48, pages 15–26, June 2013.

[41] I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. In *Proc. Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[42] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[43] K. VanLehn. Bugs are not enough: Empirical studies of bugs, impasses and repairs in procedural skills. *The Journal of Mathematical Behavior*, 1982.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. Intl. Conf. Neural Info. Process. Syst.*, volume 30, 2017.

[45] L. Wang, D. Zhang, L. Gao, J. Song, L. Guo, and H. T. Shen. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[46] Z. Wang, A. Lan, and R. Baraniuk. Mathematical formula representation via tree embeddings. Online: `https://people.umass.edu/~andrewlan/papers/preprint-forte.pdf`, 2021.

[47] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu. Handwritten mathematical expression recognition via paired adversarial learning. *International Journal of Computer Vision*, pages 1–16, 2020.

[48] Z. Xie and S. Sun. A goal-driven tree-structured neural model for math word problems. In *Proc. Int. Joint Conf. Artificial Intell.*, pages 5299–5305, 7 2019.

[49] W. Zhong, S. Rohatgi, J. Wu, C. Giles, and R. Zanibbi. Accelerating substructure similarity search for formula retrieval. In *Proc. European Conf. Info. Retrieval*, pages 714–727, 2020.

[50] W. Zhong and R. Zanibbi. Structural similarity search for formulas using leaf-root paths in operator subtrees. In L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, editors, *Proc. Intl. Conf. Neural Info. Process. Syst.*, pages 116–129, 2019.

[51] Y. Zou and W. Lu. Text2Math: End-to-end parsing text into math expressions. In *Proc. Conf. Empirical Methods Natural Lang. Process. and Intl. Joint Conf. Natural Lang. Process.*, pages 5327–5337, Nov. 2019.

Which Hammer should I Use? A Systematic Evaluation of Approaches for Classifying Educational Forum Posts

Lele Sha
Centre for Learning Analytics
Faculty of Information Technology
Monash University, Australia
lele.sha1@monash.edu

Mladen Raković
Centre for Learning Analytics
Faculty of Information Technology
Monash University, Australia
mladen.rakovic@monash.edu

Yuheng Li
Faculty of Engineering and Information Technology
University of Melbourne, Australia
yuhengleeeee@gmail.com

Alexander Whitelock-Wainwright
Centre for Learning Analytics
Portfolio of the Deputy Vice-Chancellor Education
Monash University, Australia
alex.wainwright@monash.edu

David Carroll
Centre for Learning Analytics
Portfolio of the Deputy Vice-Chancellor Education
Monash University, Australia
david.carroll@monash.edu

Dragan Gašević
Centre for Learning Analytics
Faculty of Information Technology
Monash University, Australia
dragan.gasevic@monash.edu

Guanliang Chen*
Centre for Learning Analytics
Faculty of Information Technology
Monash University, Australia
guanliang.chen@monash.edu

ABSTRACT

Classifying educational forum posts is a longstanding task in the research of Learning Analytics and Educational Data Mining. Though this task has been tackled by applying both traditional Machine Learning (ML) approaches (e.g., Logistics Regression and Random Forest) and up-to-date Deep Learning (DL) approaches, there lacks a systematic examination of these two types of approaches to portray their performance difference. To better guide researchers and practitioners to select a model that suits their needs the best, this study aimed to systematically compare the effectiveness of these two types of approaches for this specific task. Specifically, we selected a total of six representative models and explored their capabilities by equipping them with either extensive input features that were widely used in previous studies (traditional ML models) or the state-of-the-art pre-trained language model BERT (DL models). Through extensive experiments on two real-world datasets (one is open-sourced), we demonstrated that: (i) DL models uniformly achieved better classification results than traditional ML models and the performance difference ranges from 1.85% to 5.32% with respect to different evaluation metrics; (ii) when applying traditional ML models, different features should be explored and engineered to tackle different classification tasks; (iii) when applying DL models, it tends to be a promising approach to adapt BERT to the specific classification task by fine-tuning its model parameters. We have publicly released our code at `https://github.com/lsha49/LL_EDU_FORUM_CLASSIFIERS`

Keywords

Educational Forum Posts, Text Classification, Deep Neural Network, Pre-trained Language Models

1. INTRODUCTION

In the past two decades, researchers have developed a number of online educational systems to support learning, e.g., Massive Open Online Courses, Moodle, and Google Classroom. Though being widely recognized as a more flexible option compared to campus-based education, these systems are often limited by their asynchronous mode of delivery that may hinder effective interaction between instructors and students and between students themselves [27, 20]. As a remedy, the discussion forum component is often included to support communication between instructors and classmates, so students can create posts for different purposes, e.g., to ask questions, express opinions, or seek technical help. Moreover, in certain cases, instructors rely heavily on the use of a discussion forum to promote peer-to-peer collaboration, e.g., specifying a topic to spur discussions among students.

In this context, the timeliness of an instructor's response to a student post becomes critical. A group of studies has demonstrated that students' learning performance and course ex-

*Corresponding author.

Lele Sha, Mladen Rakovic, Alexander Whitelock-Wainwright, David Carroll, Dragan Gasevic and Guanliang Chen "Which Hammer should I Use? A Systematic Evaluation of Approaches for Classifying Educational Forum Posts". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 228-239. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

perience were greatly affected by the timeliness of the responses they received from instructors [2, 24, 14]. It is, therefore, critical that instructors monitor the discussion forum to provide timely help to students who need it and ensure the discussion unfolds in a way that benefits all students. However, nowadays, up to tens of thousands of students can enroll in an online course and create a variety of posts that differ by importance, i.e., not all of them warrant instructors' immediate attention. Therefore, it becomes increasingly challenging for instructors to timely identify posts that require an urgent response or to understand how well students collaborate in the discussion space.

To tackle this challenge, various computational approaches have been developed across different courses and domains to classify educational forum posts, e.g., to distinguish between urgent and non-urgent posts [2, 24] or to label posts for different levels of cognitive presence [11, 52]. Typically, these approaches relied upon traditional Machine Learning (ML) models, such as Logistic Regression, Support Vector Machine (SVM), and Random Forest. These models yielded a high level of accuracy, most often due to the extensive efforts that domain experts made to engineer input features. For post classification tasks, such features are linguistic terms describing the post content (e.g., words that represent negative emotions) and the post metadata (e.g., a creation timestamp) [31, 38, 19].

In recent years, Deep Learning (DL) models have emerged as a powerful strand of modeling approaches to tackle data-intensive problems. Compared to traditional ML models, DL models no longer requires the input of expert-engineered features; instead, they are capable of implicitly extracting such features from data with a large number of computational units (i.e., artificial neurons). Particularly, DL models have achieved great success in solving various Natural Language Processing (NLP) problems, e.g., machine translation [48], semantic parsing [22], and named entity recognition [60]. Driven by this, a few studies have been conducted and demonstrated the superiority of DL models over traditional ML models in classifying educational forum posts [24, 10, 59]. For instance, Guo et al. [24] showed that DL models can outperform a decision tree based ML model proposed in [2] by 0.1 (measured by F1 score) in terms of identifying urgent post, while [59] demonstrated that, when determining whether a post contains a question or not, the performance difference between SVM and DL models was up to 0.68 (measured by Accuracy).

Though achieving high performance, DL models have not been justified as an always-more-preferable choice compared to traditional ML models. The reasons are threefold. Firstly, studies investigating the difference in performance between traditional ML and DL models have mostly harnessed a limited set of traditional ML models for comparison, without making extensive feature engineering efforts to empower those traditional ML models. As an example, [59] compared only SVM to a group of DL models, and the SVM model in this study incorporated only one type of features, i.e., the term frequency–inverse document frequency (TF-IDF) score of the words in a post. This implies that the potential of the traditional ML models used in existing studies was not fully explored and the actual performance difference between the two types of models might be smaller than the studies to date have reported on. Secondly, researchers and practitioners often need to deliberately trade off several relevant factors before determining which model they should use in practice, and classification performance is only one of these factors. Other important factors are the availability of human-annotated training data and computing resources [29]. For instance, compared to traditional ML models, DL models demand a much larger amount of human-annotated training data, whose creation can be a time-consuming and costly process. Besides, efficient training of DL models requires access to strong computing resources (e.g., a GPU server), which may be unaffordable to researchers and practitioners with a limited budget. Most traditional ML models, on the other hand, can be easily trained on a laptop. Thirdly, the feature engineering required by traditional ML models plays an important role in contributing to a theoretical understanding of constructs that are not only useful for classification of forum posts, but are also informative about students' discussion behaviors, offering instructors insights on whether their instructional approach works as expected [45, 12, 58].

To assist researchers and educators select relevant models for post classification, this study aims at providing a systematic evaluation of the mainstream ML and DL approaches commonly used to classify educational forum posts. Throughout this evaluation, we advance research in the field by ensuring that: (i) sufficient effort is allocated to design as many meaningful features as possible to empower traditional ML models; (ii) an adequate number of representative ML and DL models is included; (iii) the effectiveness of selected models is examined by using more than one dataset, thus adding to the robustness of our approach to different educational contexts; (iv) all models are compared in the same experimental setting, e.g., with same training/test data splits, and performance reported on widely-used evaluation metrics to provide common ground for model comparison; and (v) the coding schemes used labeling discussion posts are made publicly available to motivate the replication of our study. Formally, the evaluation was guided by the following two **R**esearch **Q**uestions:

RQ1 To what extent can traditional ML models accurately classify educational forum posts?

RQ2 What is the performance difference between traditional ML models and DL models in classifying educational forum posts?

To answer the RQs, we chose two human-annotated datasets collected at two educational institutions: Stanford University and `Monash University`. We further conducted the evaluation as per the following two classification tasks: (i) whether a post requires an urgent response or not; and (ii) whether the post content is related to knowledge and skills taught in a course. Specifically, to answer RQ1, we first surveyed relevant studies that reported on applying traditional ML models to classify educational forum posts. We hence selected four models that were commonly utilized, i.e., Logistics Regression, Naïve Bays, SVM, and Random Forest. In particular, we collected features frequently employed

in the reviewed studies and incorporated them as an input to empower the four traditional ML models in our experiment. Given that these features may play different roles in different classification tasks, we further conducted a feature selection analysis to shed light on the features that must be included in the future application of these models for similar classification tasks.

To answer RQ2, we selected the two widely-adopted DL models, Convolutional Neural Network coupled with Long Short-Term Memory (CNN-LSTM) and Bi-directional LSTM (Bi-LSTM), and compared them to the four selected traditional ML models. Recent studies in DL suggested that the performance of a model adopted for solving an NLP task (CNN-LSTM or Bi-LSTM in our case, denoted as the *task model* for simplicity) can be greatly improved with the aid of state-of-the-art pre-trained language models like BERT [16] in two ways. Firstly, BERT can be used to transform the raw text of a post into a set of semantically accurate vector-based representations (i.e., *word embedding*), which comprise the input information for the task model and enable the model to distinguish among multiple characteristics of a post. Secondly, BERT can adapt itself to capture the unique data characteristics of the task at hand. To this end, BERT couples with the task model and learns the model parameters. In particular, such flexibility has been demonstrated as extremely helpful in the contexts where training data was not sufficient. Therefore, we explored the effectiveness of BERT in empowering the two DL models selected for the experiment. We provide details in Section 3.

Performance of the four traditional ML and two DL models were examined by four evaluation metrics commonly used in classification tasks, i.e., Accuracy, Cohen's κ, Area Under the ROC Curve (AUC), and F1 score. In summary, this study contributed to the literature of the classification of educational forum posts with the following main findings:

- Compared to other traditional ML models, Random Forest is more robust in classifying educational forum posts;
- Both textual and metadata features should be engineered to empower traditional ML models;
- Different features should be designed when applying traditional ML models for different classification tasks;
- DL models tend to outperform traditional ML models and the performance difference ranges from 1.85% to 5.32% with respect to different evaluation metrics;
- Using the pre-trained language model BERT benefits the performance of DL models.

2. RELATED WORK

2.1 Content Analysis of Forum Posts

Across disciplines, educators widely utilize online discussion forums to accomplish different instructional goals. For instance, instructors often provide an online discussion board as a platform for students to ask questions and get answers about course content [12, 57], argue for/against a particular issue and, in that way, engage deeply with course topics [43, 42] or work collaboratively on a course project [49, 13]. In this process, instructors monitor student involvement by reading their posts. At the same time, instructors judge student contributions in the discussion task, e.g., whether students asked a question that relates to course content vs. a question about semester tuition; described their feelings about the discussed problem vs. just rephrased the problem; or clearly communicated their ideas to classmates in a collaborative learning task. Upon identifying posts that do not contribute to the forum at the expected level, the instructor may intervene accordingly. Sometimes, such an intervention needs to be provided immediately (e.g., in a case of a post pointing out the error in the practice exam key).

With the increasing popularity of online discussion forums in the instructional context, educational researchers have become interested in conducting content analysis of students' posts to find evidence and extent of learning processes that instructors aimed to elicit in online discussion. To this end, researchers utilize coding scheme, a predefined protocol that categorizes and describes participants' behaviors representative of the observed educational construct [47, 37], e.g., knowledge building [23, 35], critical thinking [39, 35], argumentative knowledge construction [55], interaction [26, 43], social cues, cognitive/meta-cognitive skills and knowledge, depth of cognitive processing [26, 25], and self-regulated learning in collaborative learning settings [50]. As per the analytical procedure, researchers read student postings and apply a code over a unit of analysis that can be determined physically (e.g., entire post), syntactically (e.g., paragraph, sentence) or semantically (e.g., meaningful unit of text) [15, 47]. Content analysis clearly demonstrated a potential to capture relevant, fine-grained discussion behaviors and provide researchers and educators with warranted inferences made from coding data [46, 47, 28].

Manual content analysis is time-consuming [25], especially in high-enrollment courses with thousands of discussion posts that students create. To automate the process of content analysis and support monitoring of student discussion activity, various computational approaches have been developed for post classification. These approaches relied upon traditional ML models and DL models and handled four common types of post classification tasks: content, confusion, sentiment, and urgency. Below, we expand upon the studies that reported on these tasks.

2.2 Traditional Machine Learning Models

Educational researchers have applied traditional ML models to automate content analysis of online discussion posts for different instructional needs. The ML models we identified in this review are predominantly based on supervised learning paradigm and can be categorized into four general methodological approaches: regression-based (e.g., Logistics Regression [1, 61, 57, 2, 62, 36]), Bayes-based (e.g., Naive Bayes, [5, 4, 36]), kernel-based (e.g., SVM [45, 12, 5, 18, 36, 58, 40, 30]), and tree-based (e.g., Random Forest [5, 2, 36, 31, 19, 38]). These models were designed to predict an outcome variable that represented the meaning of discussion posts across different categories such as confusion, sentiment or urgency. For instance, [12] created an SVM classifier to differentiate between content-related and non-

content-related questions in a discussion thread to help instructors more easily detect content-related discourse across an extensive number of student posts in MOOC, while [1] implemented a Logistic Regression classifier to detect confusion in students' posts and automatically recommend task-relevant learning resources to students who need it. [58] applied SVM to detect student achievement emotions [41] in MOOC forums and studied the effects of those emotions on student course engagement.

In recent years, researchers became increasingly interested in analyzing the expression of urgency (e.g., regarding course content, organization, policy) in a discussion post [2]. For example, [2] developed multiple ML classifiers to identify posts that need prompt attention from course instructors. While researchers mostly implemented supervised ML models, here we also note a small group of studies that reported on using unsupervised methods to classify forum posts, e.g., a lexicographical database of sentiments [36] and minimizing entropy [8].

Traditional ML models built upon textual and non-textual features extracted from students' posts. Textual features characterize content of the discussion post, e.g., presence of domain specific words [61, 44], presence of words reflective of psychological processes [31, 38, 19], term frequency [2, 5], emotional and cognitive tone [12, 57, 40, 58, 7, 34, 1], presence of predefined hashtags [21], text readability index [62], text cohesion metrics [31, 38, 19], and measures of similarity between message text [31, 38, 19]. Non-textual features, on the other hand, include post metadata, e.g., popularity views, votes and responses [12, 45, 36], number of unique social network users [45, 1, 18], timestamp [45, 36], type (post vs. response) [36], variable that signals whether the issue has been resolved or not [61], the relative position of the most similar post [51], variable that signals whether the author of the post is also the initiator of the thread [51], page rank value of the author of current post [51], indicator if a message is the first or last in a thread [38, 31, 19], and structure of the discussion thread [53, 31, 19, 38].

Researchers computed a variety of evaluation metrics to assess performance of these models. Classification accuracy was commonly applied in studies that we reviewed (e.g., [12, 61, 36]). Generally, models achieved classification accuracy of 70% to 90% in classifying forum posts across different levels of content identification, urgency, confusion, and sentiment. We also note that some authors opted for different or additional evaluation metrics, e.g., precision/recall [12, 1, 62], AUC [12, 18], F1 [1, 2], kappa [1, 61, 57]. Across the models, authors utilized a wide range of different validation strategies (e.g., cross validation, train/test split).

We identified two major challenges researchers should be aware of when using traditional machine learning approaches to detect relevant content, confusion, sentiment and/or urgency in a discussion forum. First, traditional machine learning approaches usually involve extensive feature engineering. In the context of post classification, a huge number of textual and non-textual features of a post is practically available to researchers. Features can be generated using different text mining approaches (e.g., dictionary-based, rule-based) and can be even produced using other classifiers (e.g.,[1]). Researchers thus often face a challenge to decide which feature subset to choose to best capture educational problems (e.g., off-topic posting, misinterpreting the discussion task, unproductive interaction with peers) and/or learning process of interest (e.g., knowledge building, critical thinking, argumentation). For this reason, domain and learning experts, including course instructors, learning scientists, and educational psychologists are often needed to define a feature space that aligns with the purpose of an online discussion. Second, works in [12, 57, 4, 2] took a variety of different approaches to validate the classifiers they developed in terms of metrics, datasets, and training parameters which makes it hardly possible to directly compare the performance of these ML models.

2.3 Deep Learning Approaches

To our knowledge, relatively fewer studies attempted to explore the effectiveness of DL approaches in classifying educational forum posts [54, 59, 10, 24, 8, 3, 6]. The DL models adopted by these studies, typically, relied on the use of CNN, LSTM, or a combination of them. For instance, [54] developed a DL model called ConvL, which first used CNN to capture the contextual features that are important to discern the type of a post, and then applied LSTM to further utilize the sequential relationships between these features to assign a label to the post. Through extensive experiments, ConvL was demonstrated to achieve about 81%∼87% Accuracy in classifying discussion posts of different levels of urgency, confusion, and sentiments. In a similar vein, [59] proposed to use Bi-LSTM to better make use of the sequential relationships between different terms contained in a post (i.e., from both of the forward and backward directions). By comparing with SVM and a few DL models, this study showed that Bi-LSTM performed the best in determining whether a post contained a question or not (72%∼75% Accuracy).

It is worth noting that the success of DL models often depends on the availability of a large-amount human-annotated data for model training (typically tens of thousands at least). This, undoubtedly, limits the applicability of DL models in tackling tasks with only a small amount of training data (e.g., a few thousand). Fortunately, with the aid of pre-trained language models like BERT [16], we can still exploit the power of DL models [10]. Pre-trained language models aim to produce semantically meaningful vector-based representations of different words (i.e., word embeddings) by training on a large collection of corpora. For instance, BERT was trained on English Wikipedia articles and Book Corpus, which contain about 2,500 million and 800 million words, respectively. Two distinct benefits were brought by such pre-trained language models: (i) the word embeddings produced by them encode a rich contextual and semantic information of the text and can be well utilized by a task model (e.g., ConvL described above) to distinguish different types of input data; and (ii) a pre-trained language model can be adapted to a specific task by concatenating itself to the task model and further fine-tuning/learning their parameters as a whole with a small amount of training data. For example, [10] showed that BERT was able to boost classification Accuracy up to 83%∼92% when distinguishing posts of different levels of confusion, sentiment, and urgency.

Though gaining some impressive progress, the studies de-

Table 1: The features used as input for traditional ML models. The features used to train models are denoted as *Yes* under the column *Included*

Category	Feature	Description	# features	Studies used this feature	Included
Textual	# unigrams and bigrams	Only the top 1000 most frequent unigram/bigrams are included.	2000	[12, 57, 40, 2, 56, 62, 51]	Yes
	Post length	# words contained in a post.	1	[58, 45, 36, 62]	
	TF-IDF	The term frequency-inverse document frequency (TF-IDF) of the top 1000 most frequent unigrams.	1000	[2, 5]	
	Automated readability index	A score $\in [0, 100]$ specifying the post readability.	1	[62]	
	LIWC	A set of features denoted as scores $\in [0, 100]$ indicating the characteristics of a post from various textual categories including: *language summary*, *affect*, *function words*, *relativity*, *cognitive process*, *time orientation*, *punctuation*, *personal concerns*, *perceptual process*, *grammar*, *social* and *drives*.	84	[2, 58, 7, 34, 31, 38, 19]	
	Word overlap	The fraction of words that appeared previously in the same post thread.	-	[7]	No
	# domain-specific words	Words selected by expert to characterize a specific subject, e.g., "*equation*" and "*formula*" for Math.	-	[61, 44]	
	LDA-identified words	Words that are specific to topics discovered by applying the topic modeling method Latent Dirichlet allocation.	-	[62, 4, 44]	
	Coh-Metrix	A set of features indicating text coherence (i.e., co-reference, referential, causal, spatial, temporal, and structural cohesion) linguistic complexity, text readability, and lexical category.	-	[31, 38]	
	LSA similarity	A score indicating the average sentence similarity within a message.	-	[31]	
	Hashtags	Hashtags pre-defined by instructors to characterize the type of a post, e.g., #help and #question for confusion detection.	-	[21]	
Metadata	# views	The number of views that a post received.	1	[45, 12, 36, 62, 18]	Yes
	Anonymous post	A binary label to indicate whether a post is anonymous to other students.	1	[45, 1, 18]	
	Creation time	The day and the time when a post was made.	2	[45, 36, 18]	
	# votes	The number of votes that a post received.	1	[45, 12, 36, 62, 18]	
	Post type	A binary label to indicate whether a post is a response to another post.	1	[36, 2]	
	Response time	The amount of time before a post was responded.	-	[18]	No
	# responses	The number of responses that a post received.	-	[45, 12, 36, 18]	
	Discussion status	A binary label to indicate whether the issue has been resolved or not.	-	[61]	
	Comment Depth	A number assigned to a post to indicate its chronological position within a discussion thread.	-	[53]	
	First and Last Post	A binary label to indicate whether the post is the first or the last in a discussion thread respectively.	-	[51, 53]	

scribed above were often limited in providing a systematic comparison between the proposed DL models and existing traditional ML models. In other words, these studies either did not include traditional ML models for comparison [10, 54] or only compared DL models with only one or two traditional ML models and the potential of these traditional ML models might be suppressed due to a limited amount of efforts spent in feature engineering [59]. This necessitates a systematic evaluation of the two strands of approaches so as to better guide researchers and practitioners in selecting models for classifying educational forum posts.

3. METHODS

We open this section by describing the datasets used in our study. Then, we introduce the representative traditional ML models, including the set of features we engineered to empower those models (RQ1), and then describe the two DL models we chose to compare to the four traditional ML models (RQ2).

3.1 Datasets

To ensure a robust comparison between traditional ML and DL models in classifying educational forum posts, we adopted two datasets in the evaluation, briefly describe below.

Stanford-Urgency consists of 29,604 forum posts collected from eleven online courses at Stanford University. These courses mainly cover subjects like medicine, education, humanities, and sciences. To our knowledge, this dataset is one of the few open-sourced datasets for classifying educational forum posts and was widely used in previous studies [57, 2, 5, 10, 56, 24, 21]. In particular, Stanford-Urgency contains three types of human-annotated labels, including the degree of urgency of a post to be handled by an instructor, the degree of confusion expressed by a student in a post, and the sentiment polarity of a post. In line with the increasing research interest in detecting urgent posts [2, 10, 24, 3], we used Stanford-Urgency and focused on determining the levels of urgency of posts in this study. The count of urgent and non-urgent posts is 6,418 (22%) and 23,186 (78%), respectively. Originally, the urgency label was assigned on a Likert scale of $[1, 7]$, with 1 denoting being not urgent at all and 7 denoting being extremely urgent, respectively. Similar to previous studies [2], we pre-processed the data by treating those of value larger than or equal to 4 as urgent posts and those less than 4 as non-urgent posts, and the classification task became a binary classification problem. It is worth pointing out two notable benefits of including Stanford-Urgency: (i) the large number of posts contained in Stanford-Urgency provided sufficient training data for DL models; and (ii) in addition to the text contained in a post, Stanford-Urgency contains rich metadata information about the post, e.g., the creation time of a post, whether the creator of a post was anonymous to other students, the number of up-votes a post received, which enabled us to explore the predictive utility of different types of data.

Moodle-Content was collected by `Monash University`, the dataset contains 3,703 forum posts that students generated in the Learning Management System `Moodle` during their coursework in courses like arts, design, business, economics, computer science, and engineering. The posts were first manually labelled by a junior teaching staff and then independently reviewed (and corrected if necessary) by two additional senior teaching staff to ensure the correctness of the assigned labels. In contrast to Stanford-Urgency, this dataset contains labels to indicate whether a post was related to the knowledge and skills taught in a course or not, e.g., "*What is poly-nominal regression?*" (relevant to course content) vs. "*When is the due date to submit the second assignment?*" (irrelevant). The count of content-relevant and content-irrelevant posts is 2,339 (63%) and 1,364 (37%), respectively. Therefore, similar to the adoption of Stanford-Urgency, we also tackled a binary classification problem here. However, it should be noted that, compared to Stanford-Urgency, the metadata of posts were not available in Moodle-Content.

3.2 Traditional Machine Learning Models

Model Selection. To ensure our evaluation is systematic, we included representative models that emerged in previous studies. As summarized in Section 2.2, the traditional ML models commonly investigated to date can be roughly grouped into four categories, i.e., *regression-based*, *Bayes-based*, *kernel-based*, and *tree-based*. Therefore, we selected one model from each group and explored their capabilities in classifying educational forum posts, namely Logistics Regression, Naïve Bayes, SVM, and Random Forest.

Feature Engineering. Different from previous studies [59, 5], we argued that traditional ML models should involve an extensive set of meaningful features to fully unleash their predictive potential before being compared to DL models, specifically, we expected that ML models demonstrate improved performance when utilising more features. Therefore, we surveyed studies that reported on applying traditional ML models to classify educational forum posts, engineered features following previous studies and incorporated those features into the four traditional ML models, as summarized in Table 1. These features can be classified into two broad categories: (i) *textual* features that are extracted from the raw text of a post with the aid of NLP techniques; and (ii) *metadata* features about a post. As the metadata of posts was not available in Moodle-Content, only textual features were engineered for this dataset, while both textual and metadata features were engineered for Stanford-Urgency. We excluded several types of features from the evaluation, mainly due to the unavailability of the data required to engineer those features, e.g., *# domain-specific words*, and *Hashtags*. As for *LDA-identified words*, *Coh-Metrix*, and *LSA similarity*, we have left these features to be explored in our future work.

Feature Importance Analysis. Previous studies [12, 57, 56] have demonstrated the benefits of feature importance analysis in providing a theoretical understanding of the underlying constructs that are useful to classify educational forum posts, e.g., identifying features that are useful across different classification tasks. Therefore, we adopted the following approach to identify the top k most important features of an ML model:

1. the Chi-squared statistics between engineered features and the target classification labels were computed;
2. each time, the feature of the highest Chi-squared statistic was fed into the model and the feature was kept in the set of input features only if the classification performance had increased;
3. we repeated (2) until k most important features were identified.

3.3 Deep Learning Models

Existing studies on developing DL models to characterize different types of forum posts, typically, involved the use of CNN or LSTM, which motivated us to include the following two DL models to our evaluation:

- **CNN-LSTM** [54, 24, 59]. This model consists of: (i) an input layer, which learns an embedding representation for each word contained in the input test; (ii) a CNN layer, which performs a one-dimensional convolution operation on the embedding representation produced by the input layer and captures the contextual

information related to each word; (iii) an LSTM layer, which takes the output of the CNN layer to make use of the sequential information of the words; and (iv) a classification layer, which is fully-connected layer taking the output of the LSTM layer as input to assign a label to the input text.

- **Bi-LSTM** [59, 24, 6]. Though LSTM has been demonstrated as somewhat effective in utilizing the sequential information of long input text, they are limited in only using the previous words to predict the later words in the input text. Therefore, Bi-directional LSTM was proposed, which consists of two LSTM layers, one representing text information in the forward direction and the other in the backward direction to better capture the sequential information between different words. Formally, this model consists of: (i) an input layer (same as CNN-LSTM); (ii) a Bi-LSTM layer; and (iii) a classification layer (same as CNN-LSTM).

Both CNN-LSTM and Bi-LSTM use an input layer to learn the representation of the input text, i.e., embeddings of the words in a post. Instead of learning word embeddings during training, previous studies [17, 33, 32] suggested that pre-trained language models like BERT can be used to initialize embeddings. Such embedding initialization has been demonstrated as an effective way to facilitate a task model to acquire better performance. Therefore, we adopted BERT to initialize the input layer of both CNN-LSTM and Bi-LSTM and, correspondingly, the implemented models are denoted as Emb-CNN-LSTM, and Emb-Bi-LSTM, respectively.

In addition to word embeddings initialization, as suggested in recent studies in the field of NLP [17, 33], we can further couple BERT with a task model (i.e., CNN-LSTM or Bi-LSTM) and adapt BERT to suit the unique characteristics of a task by training BERT and the task model as a whole. In other words, the task model is often concatenated on top of BERT's output for the `[CLS]`, which is a special token used in BERT to encodes the information of the whole input text. The co-training of BERT and the task model enables BERT to fine-tune its parameters to produce task-specific word embeddings for the input text, which further facilitates the task model to determine a suitable label for the input. In fact, this fine-tuning strategy, compared to being used for embedding initialization, has been demonstrated as a more promising approach to make use of BERT. For instance, [10] showed that, even by simply coupling with a classification layer (i.e., the last layer of CNN-LSTM and Bi-LSTM), BERT was capable of accurately classifying 92% forum posts. Most importantly, it should be noted that the parameters of the coupled model can be well fine-tuned/learned with only a few thousand data samples. That means, this fine-tuning strategy enables CNN-LSTM and Bi-LSTM to be also applicable to tasks that deal with only a small amount of data, e.g., Moodle-Content in our case. In summary, we fine-tuned BERT after coupling it with CNN-LSTM (CNN-LSTM-Tuned) and Bi-LSTM (Bi-LSTM-Tuned), respectively. Besides, to gain a clear understanding of the effectiveness of this fine-tuning strategy, we coupled BERT with only a single classification layer (denoted as SCL-Tuned) and compared it with CNN-LSTM-Tuned and Bi-LSTM-Tuned. Table 2 provides a summary of the DL models implemented in this study.

Table 2: The DL models used in this study. Here, SCL denotes Single Classification Layer.

Models	Usage of BERT		Task Model		
	Embedding Initialization	Fine-tuning	CNN-LSTM	Bi-LSTM	SCL
Emb-CNN-LSTM	√		√		
Emb-Bi-LSTM	√			√	
CNN-LSTM-Tuned		√	√		
Bi-LSTM-Tuned		√		√	
SCL-Tuned		√			√

3.4 Experiment Setup

Data pre-processing. Training and testing data were randomly split in the ratio of 8:2. The Python package `NLTK` was applied to perform lower casing and stemming on the raw text of a post after removing the stop words.

Evaluation metrics. In line with previous works in classifying educational forum posts, we adopted the following four metrics, i.e., Accuracy, Cohen's κ, AUC, and F1 score, to examine model performance. We ran each model three times and reported the averaged results.

Model implementation and training. The traditional ML models (i.e., Logistics Regression, Naïve Bays, SVM, and Random Forest) were implemented with the aid of the Python package `scikit-learn` and their parameters were determined by applying grid search and fit the grid to the training data. Note all model hyper-parameters will be documented in the released GitHub repository. The ML models were trained with textual and metadata features for the Stanford-Urgency dataset, and trained with textual features for the Moodle-Content dataset. When applying the method detailed in 3.2 to perform feature importance analysis, we used F1 score as the metric to measure the changed model performance. For both CNN-LSTM and Bi-LSTM, the model parameters are selected to be comparable with similar previous works in [54, 24, 59, 10]. To this purpose, the size of the BERT embeddings used in the input layer was 768 and the number of hidden units used in the final classification layer was 1. We used the activation function sigmoid and L2 regularizer. In CNN-LSTM, the CNN layer was set to have 128 convolution filters with filter width of 5, while the LSTM layer was set to have 128 hidden states and 128 cell states. In Bi-LSTM, the number of the hidden states and cell states in the LSTM cells was both set to 128. For all DL models, (i) 10% of the training data was randomly selected as the validation data; (ii) the batch size was set to 32 and the maximum length of the input text was set to 512; (iii) the optimization algorithm Adam was used; (iv) the learning rate was set by applying the one cycle policy with maximum learning rate of 2e-05; (v) the dropout probability was set to 0.5; and (vi) the maximum number of training epochs was 50 and early stopping mechanisms were used when the model performance on the validation data starts to decrease, and data shuffling was performed at the end of each epoch. The best model is selected based on validation error. For BERT, we used the service provided by `Bert-as-service`[1].

[1]https://github.com/hanxiao/bert-as-service

Table 3: The performance of traditional ML models. The results in bold represent the best performance in each task.

	Stanford-Urgency				Moodle-Content			
Methods	**Accuracy**	**Cohen's κ**	**AUC**	**F1**	**Accuracy**	**Cohen's κ**	**AUC**	**F1**
Naïve Bays	0.7536	0.5071	0.7762	0.7844	0.7183	0.4736	0.7210	0.6870
SVM	0.8627	0.7347	0.8630	0.8185	0.7536	0.5900	0.7536	0.7530
Random Forest	**0.8915**	**0.7892**	**0.8916**	**0.8918**	**0.7544**	**0.5927**	**0.7551**	**0.7661**
Logistic Regression	0.8068	0.6287	0.8068	0.7638	0.7339	0.5251	0.7357	0.7547

Table 4: The performance of Random Forest on Stanford-Urgency when using different types of features as input. The results in bold represent the best performance.

Types of Features	**Accuracy**	**Cohen's κ**	**AUC**	**F1**
Textual	0.8639	0.7368	0.8642	0.8652
Metadata	0.8150	0.6442	0.8152	0.8136
Textual + Metadata	**0.8915**	**0.7892**	**0.8916**	**0.8918**

Table 5: The performance of Random Forest when only using the top-10 most important features (Table 6) as input. The fractions within brackets indicate the decreased performance compared to those with all available features as input (Table 3).

Stanford-Urgency				Moodle-Content			
Accuracy	**Cohen's κ**	**AUC**	**F1**	**Accuracy**	**Cohen's κ**	**AUC**	**F1**
0.8610 (-3.42%)	0.7315 (-7.31%)	0.8617 (-3.35%)	0.8628 (-3.25%)	0.7175 (-4.89%)	0.5577 (-5.91%)	0.7186 (-4.83%)	0.7358 (-3.96%)

4. RESULTS

Results on RQ1. The performance of the four traditional ML models is presented in Table 3. Across both classification tasks, Random Forest achieved the best performance, as per the calculated evaluation metrics, followed by SVM and Logistics Regression. Naïve Bayes, on the other hand, achieved the lowest performance. Specifically, Random Forest was capable of accurately classifying almost 90% of the forum posts in Stanford-Urgency, and reached an AUC and F1 score of 0.8916 and 0.8918, respectively. Besides, Cohen's κ score achieved by Random Forest for the same dataset was 0.7892, which indicates a substantial (and almost perfect) classification performance. In terms of classifying Moodle-Content, we noticed the overall performance of all models was lower than in Stanford-Urgency. This may be attributed to the lack of metadata features and significantly fewer posts in Moodle-Content than in Stanford-Urgency, making it harder for the models to reveal characteristics of different types of posts in Moodle-Content. Still, Random Forest achieved an overall accuracy, AUC, and F1 score of 0.7544, 0.7551, and 0.7661, respectively, and Cohen's κ score was very close to 0.6, which indicates an almost substantial classification performance.

Before delving into the identification of the most predictive features, we submitted each group of the textual and metadata features to the best-performing ML model (i.e., Random Forest) to depict their overall predictive power. The results are given in Table 4, derived only from Stanford-Urgency due to the unavailability of the metadata features in Moodle-Content. We observe that both textual and metadata features were useful in boosting classification performance, and textual features seem to have had a stronger capacity in distinguishing urgent from non-urgent posts. For instance, when only taking textual features into consideration, the AUC score was 0.8462, which is about 6% higher than that of metadata features (0.8152) and only 5% lower than that when considering both textual features and metadata features.

To gain a deeper understanding of the predictive power of different features, we further applied the method described in Section 3.2 to select the top 10 most important features in both Stanford-Urgency and Moodle-Content, described in Table 6. Here, several interesting observations can be made.

Firstly, almost all of the identified features were textual features, with only one exception observed in Stanford-Urgency, i.e., the metadata feature *# views*. This is in line with the findings we observed in Table 4, i.e., compared to metadata features, textual features tended to make a larger contribution in classifying forum posts. Among those textual features, we should also notice that most of them were extracted with the aid of LIWC. This corroborates with the findings presented in previous studies [31, 38, 19], i.e., LIWC is a useful tool in identifying meaningful features for characterizing educational forum posts.

Secondly, there is little overlap regarding the top ten most important features in the two tasks (only two shared feature, i.e., *LIWC: pronoun* and *LIWC: posemo*). In particular, we note that the number of features was highly related to the context of a classification task. In the Stanford-Urgency case, a number of top features were associated with a sense of stimulation (e.g., *anxiety*, *affect*, *drive*), which represents a subjective representation of urgency. In the Moodle-Content case, features were more associated with a sense of investigation (e.g., *Analytic* and *Understand*). This shows that different classification tasks (i.e., Urgency vs. Content-related) require task-specific features to best capture the task-specific information (i.e., whether the post expressed a sense of ur-

Table 6: The top 10 most important features used in Random Forest. Features shared by the two tasks are in bold.

Stanford-Urgency		Moodle-Content	
Features	**Description**	**Features**	**Description**
Metadata: # views	The number of views that a post received.	Post length	# words contained in a post.
LIWC: pronoun	# of the occurrence of all pronouns (e.g., personal and impersonal pronouns)	LIWC: Analytic	A score indicating the formal, logical, and hierarchical thinking patterns in a post
Unigram: they	# of the occurrence of the word "*they*"	LIWC: Tone	A score indicating the emotional tone conveyed in a post
LIWC: number	# of the occurrence of the digital numbers	**LIWC: pronoun**	# of the occurrence of all pronouns (e.g., personal and impersonal pronouns)
LIWC: affect	A score indicating the overal emotion (positive and negative) of a post	LIWC: ppron	# of the occurrence of all personal pronoun (e.g., he, she, me) in a post
LIWC: posemo	A score indicating the positive emotion of a post	Unigram: I	# of the occurrence of the word "*I*"
LIWC: drives	A score indicating the needs, motives, and drives of a post (e.g., references to success and failure)	**LIWC: posemo**	A score indicating the positive emotion of a post
LIWC: power	A score indicating the power of a post (e.g., reference to dominance)	TF-IDF: understand	The TF-IDF score of the word "*understand*"
LIWC: anx	A score indicating the anxiety conveyed in a post	LIWC: affiliation	A score indicating the capacity for enjoying close, harmonious relationships conveyed in a post
LIWC: QMark	# of the occurrence of question mark	LIWC: Exclam	# of the occurrence of exclamation mark

Table 7: The performance of DL models. The results in bold represent the best performance in each task. The fractions within brackets indicate the increased performance compared to the best performance achieved by Random Forest (Table 3).

	Models	**Accuracy**	**Cohen's κ**	**AUC**	**F1**
Stanford-Urgency	1. Emb-CNN-LSTM	0.9203 (3.23%)	0.8192 (3.80%)	0.9201 (3.20%)	0.9203 (3.20%)
	2. Emb-Bi-LSTM	0.9159 (2.73%)	0.8051 (2.01%)	0.9153 (2.66%)	0.9159 (2.71%)
	3. CNN-LSTM-Tuned	**0.9211** (3.32%)	**0.8210** (4.02%)	**0.9221** (3.42%)	**0.9221** (3.40%)
	4. Bi-LSTM-Tuned	0.9210 (3.30%)	0.8196 (3.85%)	0.9208 (3.28%)	0.9210 (3.27%)
	5. SCL-Tuned	0.9210 (3.31%)	0.8206 (3.98%)	0.9215 (3.35%)	0.9219 (3.38%)
Moodle-Content	6. CNN-LSTM-Tuned	**0.7934** (5.17%)	**0.6230** (5.11%)	**0.7952** (5.32%)	**0.7993** (4.33%)
	7. Bi-LSTM-Tuned	0.7854 (4.11%)	0.6220 (4.93%)	0.7901 (4.64%)	0.7913 (3.29%)
	8. SCL-Tuned	0.7716 (2.29%)	0.6092 (2.77%)	0.7733 (2.42%)	0.7803 (1.85%)

gency).

Moreover, when solely using the top 10 features as an input, the performance of Random Forest was 3.25%∼7.31% lower than the performance obtained after incorporating all available features (Table 5). This finding hence confirms that while the traditional ML models can achieve good classification performance using only the top 10 best features, there is still potential for improvement when using more features. Hence, researchers should attempt to apply more features to fully unleash traditional ML models' capability.

Results on RQ2. The performance of the implemented DL models is presented in Table 7. As Moodle-Content contained only 3,703 labeled posts, that was likely to be insufficient to support the training of CNN-LSTM or Bi-LSTM from scratch. Therefore, we only implemented the fine-tuned models, i.e., CNN-LSTM-Tuned, Bi-LSTM-Tuned, and SCL-Tuned on Moodle-Content. Several observation can be derived based on the results in Table 7.

Firstly and unsurprisingly, DL models uniformly achieved a better performance than traditional ML models. This corroborates findings reported in [59, 54, 33, 24]. DL models are, therefore, superior to traditional ML models in terms of capturing the characteristics of a dataset and obtaining better classification results. However, we should note that the performance difference between traditional ML models and DL models was not that large. Specifically, the best-performing model CNN-LSTM-Tuned achieved an improvement of only 3.32% in Accuracy, 4.02% in Cohen's κ, 3.42% in AUC, and 3.40% in F1 score. In particular, the Cohen's κ score was 0.8210, which suggests an almost perfect classification performance.

Secondly, contrasting findings reported in [59], we found that CNN-LSTM slightly outperform Bi-LSTM in most cases (i.e., Row 1 vs. Row 2, Row 3 vs. Row 4, and Row 6 vs. Row 7 in Table 7). Thirdly, instead of using BERT for embedding initialization, the classification model would achieve better performance by fine-tuning BERT by coupling it with the task model and training the coupled model as a whole (i.e., Row 1-2 vs. Row 3-4 in Table 7), though the improvement was rather limited, e.g., less than 1% when comparing to that of Emb-CNN-LSTM and CNN-LSTM-Tuned on Stanford-Urgency. Fourthly, we showed that in Stanford-Urgency, by simply coupling BERT with a single classification layer (SCL-Tuned, Row 5 in Table 7), the classification perfor-

mance was almost as good as those derived by coupling BERT with more complex DL models like CNN-LSTM and Bi-LSTM (Row 3-4 in Table 7). This implies that, BERT can capture the rich semantic information hidden behind a post, which can be used to deliver adequate classification performance even by employing a single classification layer.

5. DISCUSSION AND CONCLUSION

The classification of educational forum posts has been a longstanding task in the research of Learning Analytics and Educational Data Mining. Though quite some previous studies have been conducted to explore the applicability and effectiveness of traditional ML models and DL models in solving this task, a systematic comparison between these two types of approaches has not been conducted to date. Therefore, this study set out to provide such an evaluation with aiming at paving the road to researchers and practitioners to select appropriate predictive models when tackling this task. Specifically, we compared the performance of four representative traditional ML models (i.e., Logistics Regression, Naïve Bays, SVM, and Random Forest) and two commonly-applied DL models (i.e., CNN-LSTM and Bi-LSTM) on two datasets. We further elaborate on several implications that our work may have on the development of classifiers for educational forum posts. We also list limitations to be addressed in future studies.

Implications. Firstly, the performance difference between traditional ML models and DL models was not as large as reported by previous studies (e.g., [59]). More specifically, we showed that traditional ML models were often inferior to DL models in terms of only 1.85% to 5.32% decrease in classification performance measured by Accuracy, Cohen's κ, AUC, and F1 score. This finding implies that, when researchers and practitioners have no access to strong computing resources and, for this reason, cannot utilize DL models, they can still achieve acceptable classification performance by using traditional ML models, as long as those ML models incorporate carefully-crafted features.

Secondly, our results demonstrate that the performance of Random Forest classifier is more robust compared to other traditional ML models. This implies that other more advanced tree-based ML models (e.g., Gradient Tree Boosting [9]) might be worth exploring to achieve even higher classification performance. Besides, given that the most important feature in Stanford-Urgency was *# views* (Table 6) and the models' performance in Moodle-Content might be suppressed due to the unavailability of metadata features, it may be worth paying special attention to acquiring and using metadata features when applying traditional ML models. Another finding suggests that little overlap was detected between the top 10 most important features selected in each of the two classification tasks (Table 6). This implies when tackling a classification task, features should be designed to suit the unique characteristics of the task and fit the theoretical model utilized to annotate data (e.g., with predefined coding scheme). This aligns with findings presented in [31, 38, 19], in different phases of cognitive presence, different importance scores were obtained for the same features. Lastly, researchers and practitioners may wish to take advantage of pre-trained language models like BERT when developing DL models. Our experiment showed that BERT can be effectively used in two ways, i.e., (i) to initialize the word embeddings of the post text as the input for a task model; or (ii) to suit the needs of the specific classification task by coupling itself with the task model and then fine-tuning model parameters. Particularly, the second way enables DL models to be applicable to tasks that deal with only a small amount of human-annotated data, like in Moodle-Content).

Limitations. Firstly, the evaluation presented in this study focused only two classification tasks, i.e., Stanford-Urgency and Moodle-Content. To further increase the reliability of the presented findings, more tasks should be included and investigated, e.g., determining the level of confusion that a student expressed in a forum post or whether the sentiment contained in the post is positive or negative [10, 54]. Secondly, a few types of features were not included when exploring the capabilities of traditional ML models in our evaluation, e.g., *# domain-specific words* and *LDA-identified words*. To accurately depict the upper bound of the performance of traditional ML models in classifying educational forum posts, it would be worthy to recruit domain experts to further engineer and make use of these features. Thirdly, we should notice that the DL models used in our evaluation (i.e., CNN-LSTM and Bi-LSTM) only utilized the raw text of a post as input and left the metadata features untapped. Given that metadata features have been demonstrated of great importance in the application of traditional ML models, future research efforts should also be allocated to design more advanced DL models that are capable of using both the raw text of a post and the metadata of the post for classification.

Lastly, we acknowledge that, due to the scope of this study, we did not attempt to investigate the reasons causing the performance difference between traditional ML models and DL models, e.g., whether the two categories of models misclassified the same types of messages. In the future, we will further investigate whether the performance difference between traditional ML models and DL models can be attributed to their model structures and explore potential methods to boost their classification performance, e.g., collecting additional forum posts to continue the pre-training of BERT before coupling it with a downstream classification model.

6. REFERENCES

[1] A. Agrawal, J. Venkatraman, S. Leonard, and A. Paepcke. Youedu: addressing confusion in mooc discussion forums by recommending instructional video clips. 2015.

[2] O. Almatrafi, A. Johri, and H. Rangwala. Needle in a haystack: Identifying learner posts that require urgent response in mooc discussion forums. *Computers & Education*, 118:1–9, 2018.

[3] L. Alrajhi, K. Alharbi, and A. I. Cristea. A multidimensional deep learner model of urgent instructor intervention need in mooc forum posts. In *Intelligent Tutoring Systems*, pages 226–236. Springer International Publishing, 2020.

[4] T. Atapattu, K. Falkner, and H. Tarmazdi. Topic-wise classification of mooc discussions: A visual analytics approach. *EDM*, 2016.

[5] A. Bakharia. Towards cross-domain mooc forum post

classification. In *Learning@Scale*, pages 253–256, 2016.

[6] F. Brahman, N. Varghese, and S. Bhat. Effective Forum Curation via Multi-task Learning. page 8, 2020.

[7] A. Caines, S. Pastrana, A. Hutchings, and P. J. Buttery. Automatically identifying the function and intent of posts in underground forums. *Crime Science*, 7(1):19, 2018.

[8] J. Chen, J. Feng, X. Sun, and Y. Liu. Co-training semi-supervised deep learning for sentiment classification of mooc forum posts. *Symmetry*, 12(1):8, 2020.

[9] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.

[10] B. Clavié and K. Gal. Edubert: Pretrained deep language models for learning analytics. *arXiv preprint arXiv:1912.00690*, 2019.

[11] A. Cohen, U. Shimony, R. Nachmias, and T. Soffer. Active learners' characterization in mooc forums and their generated knowledge. *British Journal of Educational Technology*, 50(1):177–198, 2019.

[12] Y. Cui and A. F. Wise. Identifying content-related threads in mooc discussion forums. In *Learning@Scale*, pages 299–303, 2015.

[13] D. D. Curtis and M. J. Lawson. Exploring collaborative online learning. *Journal of Asynchronous learning networks*, 5(1):21–34, 2001.

[14] M. Dascalu, S. Trausan-Matu, D. S. McNamara, and P. Dessus. Readerbench: Automated evaluation of collaboration based on cohesion and dialogism. *International journal of computer-supported collaborative learning*, 10(4):395–423, 2015.

[15] B. De Wever, T. Schellens, M. Valcke, and H. Van Keer. Content analysis schemes to analyze transcripts of online asynchronous discussion groups: A review. *Computers & education*, 46(1):6–28, 2006.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] J. Dong, F. He, Y. Guo, and H. Zhang. A commodity review sentiment analysis based on bert-cnn model. In *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, pages 143–147. IEEE, 2020.

[18] L. Feng, G. Liu, S. Luo, and S. Liu. A transferable framework: Classification and visualization of mooc discussion threads. In *International Conference on Neural Information Processing*, pages 377–384. Springer, 2017.

[19] M. Ferreira, V. Rolim, R. F. Mello, R. D. Lins, G. Chen, and D. Gašević. Towards automatic content analysis of social presence in transcripts of online discussions. In *LAK*, pages 141–150, 2020.

[20] E. L. Fu, J. van Aalst, and C. K. Chan. Toward a classification of discourse patterns in asynchronous online discussions. *International Journal of Computer-Supported Collaborative Learning*, 11(4):441–478, 2016.

[21] S. A. Geller, N. Hoernle, K. Gal, A. Segal, A. X. Zhang, D. Karger, M. T. Facciotti, and M. Igo. # confused and beyond: detecting confusion in course forums using students' hashtags. In *LAK*, pages 589–594, 2020.

[22] E. Grefenstette, P. Blunsom, N. De Freitas, and K. M. Hermann. A deep architecture for semantic parsing. *arXiv preprint arXiv:1404.7296*, 2014.

[23] C. N. Gunawardena, C. A. Lowe, and T. Anderson. Analysis of a global online debate and the development of an interaction analysis model for examining social construction of knowledge in computer conferencing. *Journal of educational computing research*, 17(4):397–431, 1997.

[24] S. X. Guo, X. Sun, S. X. Wang, Y. Gao, and J. Feng. Attention-based character-word hybrid neural networks with semantic and structural information for identifying of urgent posts in mooc discussion forums. *IEEE Access*, 7:120522–120532, 2019.

[25] N. Hara, C. J. Bonk, and C. Angeli. Content analysis of online discussion in an applied educational psychology course. *Instructional science*, 28(2):115–152, 2000.

[26] F. Henri. Computer conferencing and content analysis. In *Collaborative learning through computer conferencing*, pages 117–136. Springer, 1992.

[27] K. F. Hew and W. S. Cheung. Students' and instructors' use of massive open online courses (moocs): Motivations and challenges. *Educational research review*, 12:45–58, 2014.

[28] D. H. Jonassen and H. Kwon. Communication patterns in computer mediated versus face-to-face group problem solving. *Educational technology research and development*, 49(1):35, 2001.

[29] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[30] A. Khan, I. Ibrahim, M. I. Uddin, M. Zubair, S. Ahmad, A. Firdausi, M. Dzulqarnain, and M. Zaindin. Machine learning approach for answer detection in discussion forums: An application of big data analytics. *Scientific Programming*, 2020, 2020.

[31] V. Kovanović, S. Joksimović, Z. Waters, D. Gašević, K. Kitto, M. Hatala, and G. Siemens. Towards automated content analysis of discussion transcripts: A cognitive presence case. In *LAK*, pages 15–24, 2016.

[32] X. Li, L. Bing, W. Zhang, and W. Lam. Exploiting bert for end-to-end aspect-based sentiment analysis. *arXiv preprint arXiv:1910.00883*, 2019.

[33] X. Li, H. Zhang, Y. Ouyang, X. Zhang, and W. Rong. A shallow bert-cnn model for sentiment analysis on moocs comments. In *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*, pages 1–6. IEEE, 2019.

[34] M. Lui and T. Baldwin. Classifying user forum participants: Separating the gurus from the hacks, and other tales of the internet. In *Proceedings of the Australasian Language Technology Association Workshop 2010*, pages 49–57, 2010.

[35] R. M. Marra, J. L. Moore, and A. K. Klimczak. Content analysis of online discussion forums: A comparative analysis of protocols. *Educational Technology Research and Development*, 52(2):23, 2004.

[36] P. M. Moreno-Marcos, C. Alario-Hoyos, P. J. Muñoz-Merino, I. Estévez-Ayres, and C. D. Kloos.

Sentiment analysis in moocs: A case study. In *2018 IEEE Global Engineering Education Conference (EDUCON)*, pages 1489–1496. IEEE, 2018.

[37] J. Mu, K. Stegmann, E. Mayfield, C. Rosé, and F. Fischer. The acodea framework: Developing segmentation and classification schemes for fully automatic analysis of online discussions. *International journal of computer-supported collaborative learning*, 7(2):285–305, 2012.

[38] V. Neto, V. Rolim, R. Ferreira, V. Kovanović, D. Gašević, R. D. Lins, and R. Lins. Automated analysis of cognitive presence in online discussions written in portuguese. In *Proceedings of the 13th European Conference on Technology Enhanced Learning*, pages 245–261. Springer, 2018.

[39] D. R. Newman, B. Webb, and C. Cochrane. A content analysis method to measure critical thinking in face-to-face and computer supported group learning. *Interpersonal Computing and Technology*, 3(2):56–77, 1995.

[40] A. Ntourmas, N. Avouris, S. Daskalaki, and Y. Dimitriadis. Comparative study of two different mooc forums posts classifiers: analysis and generalizability issues. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–8. IEEE, 2019.

[41] R. Pekrun. The control-value theory of achievement emotions: Assumptions, corollaries, and implications for educational research and practice. *Educational psychology review*, 18(4):315–341, 2006.

[42] R. Rabbany, S. Elatia, M. Takaffoli, and O. R. Zaïane. Collaborative learning of students in online discussion forums: A social network analysis perspective. In *EDM*, pages 441–466. Springer, 2014.

[43] M. Raković, Z. Marzouk, A. Liaqat, P. H. Winne, and J. C. Nesbit. Fine grained analysis of students' online discussion posts. *Computers & Education*, 157:103982, 2020.

[44] A. Ramesh, S. H. Kumar, J. Foulds, and L. Getoor. Weakly supervised models of aspect-sentiment for online course discussion forums. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 74–83, 2015.

[45] L. A. Rossi and O. Gnawali. Language independent analysis and classification of discussion threads in coursera mooc forums. In *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*, pages 654–661. IEEE, 2014.

[46] L. Rourke and T. Anderson. Exploring social communication in computer conferencing. *Journal of Interactive Learning Research*, 13(3):259–275, 2002.

[47] L. Rourke and T. Anderson. Validity in quantitative content analysis. *Educational technology research and development*, 52(1):5, 2004.

[48] S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi, and S. Jain. Machine translation using deep learning: An overview. In *2017 international conference on computer, communications and electronics (comptelix)*, pages 162–167. IEEE, 2017.

[49] H.-J. So. When groups decide to use asynchronous online discussions: collaborative learning and social presence under a voluntary participation structure. *Journal of Computer Assisted Learning*, 25(2):143–160, 2009.

[50] M. Sobocinski, J. Malmberg, and S. Järvelä. Exploring temporal sequences of regulatory phases and associated interactions in low-and high-challenge collaborative learning sessions. *Metacognition and Learning*, 12(2):275–294, 2017.

[51] C. Sun, S.-w. Li, and L. Lin. Thread structure prediction for mooc discussion forum. In *International Conference of Pioneering Computer Scientists, Engineers and Educators*, pages 92–101. Springer, 2016.

[52] X. Wang, D. Yang, M. Wen, K. Koedinger, and C. P. Rosé. Investigating how student's cognitive behavior in mooc discussion forums affect learning gains. *EDM*, 2015.

[53] Z. Waters, V. Kovanović, K. Kitto, and D. Gašević. Structure matters: Adoption of structured classification approach in the context of cognitive presence classification. In *Proceedings of the 11th Asia Information Retrieval Societies Conference*, pages 227–238. Springer, 2015.

[54] X. Wei, H. Lin, L. Yang, and Y. Yu. A Convolution-LSTM-Based Deep Neural Network for Cross-Domain MOOC Forum Post Classification. *Information*, 8(3):92, Sept. 2017. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[55] A. Weinberger and F. Fischer. A framework to analyze argumentative knowledge construction in computer-supported collaborative learning. *Computers & education*, 46(1):71–95, 2006.

[56] A. F. Wise, Y. Cui, W. Jin, and J. Vytasek. Mining for gold: Identifying content-related mooc discussion threads across domains through linguistic modeling. *The Internet and Higher Education*, 32:11–28, 2017.

[57] A. F. Wise, Y. Cui, and J. Vytasek. Bringing order to chaos in mooc discussion forums with content-related thread identification. In *LAK*, pages 188–197, 2016.

[58] W. Xing, H. Tang, and B. Pei. Beyond positive and negative emotions: Looking into the role of achievement emotions in discussion forums of moocs. *The Internet and Higher Education*, 43:100690, 2019.

[59] Y. Xu and C. F. Lynch. What do you want? applying deep learning models to detect question topics in mooc forum posts? In *Wood-stock'18: ACM Symposium on Neural Gaze Detection*, pages 1–6, 2018.

[60] V. Yadav and S. Bethard. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*, 2019.

[61] D. Yang, M. Wen, I. Howley, R. Kraut, and C. Rose. Exploring the effect of confusion in discussion forums of massive open online courses. In *Learning@Scale*, pages 121–130, 2015.

[62] Z. Zeng, S. Chaturvedi, and S. Bhat. Learner affect through the looking glass: Characterization and detection of confusion in online courses. *EDM*, 2017.

Acting Engaged: Leveraging Play Persona Archetypes for Semi-Supervised Classification of Engagement

Benjamin D. Nye [*]
Inst. for Creative Technologies
Univ. of Southern California
nye@ict.usc.edu

Mark G. Core [*]
Inst. for Creative Technologies
Univ. of Southern California
core@ict.usc.edu

Shikhar Jaiswal [*] [†]
Microsoft Research
Bangalore, India
t-sjaiswal@microsoft.com

Aviroop Ghosal [†]
Amazon.com, Inc.
Detroit, Michigan, USA
aviroopg41@gmail.com

Daniel Auerbach
Inst. for Creative Technologies
Univ. of Southern California
auerbach@ict.usc.edu

ABSTRACT

Engaged and disengaged behaviors have been studied across a variety of educational contexts. However, tools to analyze engagement typically require custom-coding and calibration for a system. This limits engagement detection to systems where experts are available to study patterns and build detectors. This work studies a new approach to classify engagement patterns without expert input, by using a play persona methodology where labeled archetype data is generated by novice testers acting out different engagement patterns in a system. Domain-agnostic task features (e.g., response time to an activity, scores/correctness, task difficulty) are extracted from standardized data logs for both archetype and authentic user sessions. A semi-supervised methodology was used to label engagement; bottom-up clusters were combined with archetype data to build a classifier. This approach was analyzed with a focus on cold-start performance on small samples, using two metrics: consistency with larger full-sample cluster assignments and stability of points staying in the same cluster once assigned. These were compared against a baseline of clustering without an incrementally trained classifier. Findings on a data set from a branching multiple-choice scenario-based tutoring system indicated that approximately 52 unlabeled samples and 51 play-test labeled samples were sufficient to classify holdout sessions at 85% consistency with a full set of 145 unsupervised samples. Additionally, alignment to play persona samples for the full set matched expert labels for clusters. Use-cases and limitations of this approach are discussed.

Keywords

Engagement, Machine Learning, Semi-Supervised, Clustering, Classification, Archetypes, Play Personas

1. INTRODUCTION

Engagement represents a necessary (though not sufficient) condition for learning. Engagement has been shown to impact learning [4] and persistence [9]. Research has also found that engagement is actionable and can be increased [25]. This is a particularly important topic for computer-based learning: unlike in a classroom, where engagement can be assessed and acted on by an instructor in real-time, patterns of engagement are often not visible [10].

However, building engagement analytics for a new system is time consuming. Custom metrics are typically developed and then require substantial data to identify patterns (i.e., the cold-start problem). Worse, the extensive effort to design such analytics is buried in application-specific code. While heuristics are available to infer disengagement, such as response times under 3 seconds [6], applying these to different systems requires benchmarking and calibrating detectors for the content and system. Efforts to analyze engagement often start almost from scratch. This is unfortunate, since research on behavioral engagement has identified patterns which appear to generalize across systems [3, 4, 6, 14, 15].

To address this gap, we are researching a service for analyzing and classifying engagement that relies on a standards-based learning record store [1]. This effort is called the Service for Measurement and Adaptation to Real-Time Engagement (SMART-E). Rather than being optimized to analyze a specific system or data set, SMART-E targets three high-level goals: 1) *Cold-Start Calibration*: ability to identify and benchmark engagement behaviors, which does not require large data sets or in-depth expert analysis; 2) *Re-Usability*: reliance on standards and data available from most learning environments; and 3) *Actionability*: generation of actionable insights, which an instructor or adaptive system could leverage or investigate further.

SMART-E is influenced by two techniques: 1) *semi-supervised learning*, which trains with a small set of labeled data and

[*]Denotes equal contribution.

[†]Shikhar Jaiswal and Aviroop Ghosal contributed to this research as student researchers at the University of Southern California.

Benjamin Nye, Mark G. Core, Shikhar Jaiswal, Aviroop Ghosal and Daniel Auerbach "Acting Engaged: Leveraging Player Persona Archetypes for Semi-Supervised Classification of Engagement". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 240-251. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

a larger set of unlabeled data and 2) *play persona*, behavioral archetypes commonly used for testing and analysis of video games [7, 32]. Our paper describes the process and findings from applying this approach to a data set from a scenario-based tutoring system for training counseling skills. Contributions from this work include a) reviewing features that generalized engagement analytics should consider, b) developing a pipeline for analyzing engagement which does not require expert labeling or application-specific feature engineering, and c) demonstrating the effectiveness of a semi-supervised approach with reasonable data requirements (e.g., about 50 samples each of labeled and unlabeled data) to approximate inferences that experts might make given similar data. As such, this research represents a step toward a generalized framework for diagnosing learner engagement that does not require an expert researcher analyzing data or observing subjects.

2. BACKGROUND AND THEORY

Across the learning science community, engagement is defined and measured in vastly different ways, ranging from split-second physiological responses (e.g., eye tracking, facial affect) to long-term trends lasting months or years (e.g., returning to a system, building social ties) [2, 13]. The research in this paper targets behavioral engagement at the task level (e.g., time spent working through a problem) and session level (e.g., sustained effort to improve performance and learning).

A key reason for this focus is data availability and data interpretability. Most systems collect data logs at these levels and, as described next, substantial research has also identified common behavioral patterns. Research on lower-level affective cues (e.g., facial affect) has found certain actionable events that generalize (e.g., gaze inattention [15]), but other patterns are not trivial to generalize due to differences between individuals or across contexts [28]. Moreover, facial data is often unavailable due to the privacy issues involved with recording learning. Larger time scales are not the focus of this work because engagement levels over those time scales would require longitudinal data and also are more likely to be visible to instructors (e.g., absences).

2.1 Patterns of Behavioral Engagement

Behavioral engagement analysis from log files has shown repeated evidence of useful, actionable patterns, such as response time, response time vs. accuracy/correctness interactions, approach vs. avoidance behaviors relative to problem difficulty (e.g., skipping hard problems), and noisiness of answer quality (e.g., carelessness) [5]. Response time, particularly very fast response time, is one of the most obvious features linked to behaviors associated with disengagement (e.g., guessing, skipping, straight-lining). For scored tasks, the interaction between response time and correctness has been extensively researched in the study of basic cognition as well as authentic learning tasks [6]. The relationship between correctness and time is frequently a logistic relationship (assuming that time does not directly impact scoring): with very fast responses, correctness is approximately random, increasing rapidly to better than chance for more ordinary response time, and approaching an individual skill-level asymptote as time increases. At very large times, answer quality may once again decrease, either due to distraction (e.g., multi-tasking) or difficulty selecting a final answer [27].

More complex interactions often require understanding the relative problem difficulty. Research indicates that students with poor learning outcomes tend to avoid or abuse hints on problems that they find difficult [5]. Conversely, self-regulated learners may be more likely to skip or "game" through problems that that are easier relative to their skill level but dedicate more time to harder problems [33]. While not yet investigated, this might also imply that more self-regulated learners may be less likely to demonstrate wheel-spinning [18] since they are more actively monitoring the usefulness of tasks.

Estimates of answer correctness versus expected correctness have also been used, though these are likely most clear when the learner is close to mastery. Of these, carelessness and "slips" are the most well-established mechanisms [12]. More generally, there may be value in investigating any situation where correctness appears decoupled from traditional factors (e.g., little correlation between time and answer quality, little correlation between expect mastery and later performance). However, such decoupling could be due to poor task design (e.g., item response issues [20]) or problems unrelated to engagement (e.g., attention or memory problems), so additional context may be needed to interpret this.

2.2 Archetypes for Behavioral Engagement

When considering these different patterns of behavioral engagement and disengagement, we posit that engagement has at least two dimensions: a) passiveness vs. activeness and b) avoidance vs. approach. For example, passive avoidance represents disengagement commonly associated with boredom such as distraction or skipping through material. By comparison, other learners employ short-cut strategies to cheat or cherry-pick tasks to minimize effort while still providing acceptable performance (active avoidance). A similar division exists for engaged learners, in that some study almost exclusively on assigned content (passive approach) while others monitor and self-regulate their effort to focus their learning (active approach).

These latent engagement factors may be evident through different observed patterns. For example, while distraction and racing through material both represent disengagement, their data patterns will look very different. In considering these patterns, we developed the following candidates which may be evident across a variety of systems:

- Diligent (Active Engagement): Spends somewhat more time on tasks and shows correspondingly better performance, and more likely to complete optional tasks.
- Self-Regulated (Active Engagement): Seeks out and spends greater time on harder tasks, but may skip or disengage on easier tasks. [22, 33].
- Cherry Picking (Active Disengagement): Seeks out easier tasks or abuses features to make tasks easier (e.g., hint abuse), and avoids harder tasks [3].
- Nominal Engagement (Passive Engagement): Completes tasks as recommended or assigned, with ordinary time-on-task and performance.

- Expert/Recall (Passive Engagement): Regardless of difficulty level, completes tasks very rapidly and with high performance. Possibly an expert on the content, but might also be shallow recall or lookup.

- Racing/Guessing (Passive Disengagement): Rapidly answers (potentially multiple times) despite relatively poor performance [26].

- Distracted/Slow (Passive Disengagement): Uncommonly delayed or irregular answers, particularly when extra time does not appear to improve performance [27].

As with prior research on engagement, we do not assume that these archetypes are necessarily stable for a specific user across all content, but that they represent modes of interaction during learning. Additionally, these candidate patterns are not exhaustive and the specific evidence for each pattern may not be identical: while racing through material might involve rapid guessing in one system, in another it might involve skipping material entirely. Historically, this has meant that detectors are tuned using expert-labeled observations and/or expert feature engineering.

2.3 Play Persona as a Labeling Methodology

This work applies a new approach to generating engagement labels for user sessions. While substantial research has been conducted on engagement, existing methods for determining engagement during computer-based learning are challenging to scale. Our research is intended to complement three methods currently in-use: expert observers, sensor-based affect detection, and self-report [13].

Expert observers can be trained on a specific coding manual until they reach high levels of agreement. Using techniques such as BROMP [29], a trained observer can monitor and label engagement events for multiple students. The primary barriers to collecting this data are the number of trained observers required and issues of privacy and technology (e.g., observing students in online courses). Automated affect detection (e.g, automated facial affect detection) has also been used to analyze engagement [28, 19]. While in principle facial affect scales to a large number of learners, engagement is hard to interpret without also analyzing behavioral patterns (e.g., screen recordings, log files). As with human observers, privacy issues may prevent the necessary recording of data. Moreover, for both human and automated labeling, while learner states may be recorded, they do not include any interpretation about what strategies a learner is using (e.g., focusing on hard vs. easy problems). Self-report offers a different type of engagement label. Users can report their overall engagement and may also be able to describe the learning strategies that they are using [13]. However, self-reported engagement can be affected by reporting bias (e.g., claiming to be more engaged) or subjectivity of engagement ratings.

To address these limitations, we identified play persona as a way to generate labeled engagement data. Play persona are behavioral archetypes often used for testing and analysis of video games, that reflect different goals and behavior patterns [35, 32]. For example, in a strategy game there are recognized archetypes such as the Builder (invest in long-term expansion) versus Greedy-Optimizer (take quick wins) [34]. Likewise, research on Massive Multiplayer Games (e.g., [36]) has identified behavior archetypes such as competitors who focus on head-to-head tasks and explorers who focus on exploring the world. Artificial game players can be crafted to mimic these play persona for procedural play-testing [21].

We hypothesize that play persona methods can also be useful to identify and label engagement patterns with the modification that human testers will act out these roles (e.g., diligent) which would be difficult to simulate artificially. If this approach is useful, it has at least three advantages over existing methods. First, it ensures rapid data collection of labels, since rather than having unbalanced labels (i.e., 80% of real users might be in one bin), testers can be directed to act out a variety of roles. Second, play-test labels should be interpretable since the intent of the learner is known, as opposed to purely bottom-up patterns or self-reported labels, which require experts to infer underlying strategies. Finally, despite some constraints (e.g., difficulty in faking more or less knowledge), dedicated testers may be able to play out multiple archetypes and do so repeatedly, reducing the need to recruit new testers.

3. RESEARCH QUESTIONS

This work investigates techniques to leverage play-testing data for detecting engagement patterns. However, this approach will only be feasible if testers reasonably approximate the behavior of real users. It also relies on the assumption that while systems may differ, the main engagement archetypes will be fairly predictable (e.g., some users will be highly invested in learning every piece of content, others will be trying to get through as fast as possible). In this work, we examine the feasibility of play-testing to help classify engagement patterns, and in particular investigate the following questions:

- Q1 (Distinctiveness): Are the data patterns for a set of play-tester archetypes distinct (different testers act similarly, given similar instructions)?

- Q2 (Alignment): Will play-test archetypes align with unsupervised clusters producing labeled clusters similar to how experts would label them?

- Q3 (Semi-Supervised Comparison): Will a semi-supervised approach that builds a classifier from play-test and aligned data label individual learners more consistently than relying only on bottom-up clusters?

- Q4 (Basic Features): Will average response time and scores, in simple systems, be sufficient for reasonable engagement labels?

- Q5 (Expanding Features): Will increasing the number of features to include task difficulty and feature interactions lead to greater consistency in fewer samples?

These questions investigate the strengths and limitations of the approach. Specifically, Q1 and Q2 focus on the reliability of play-test labels to label unsupervised data, as compared to human ratings. Q3 examines if building a semi-supervised

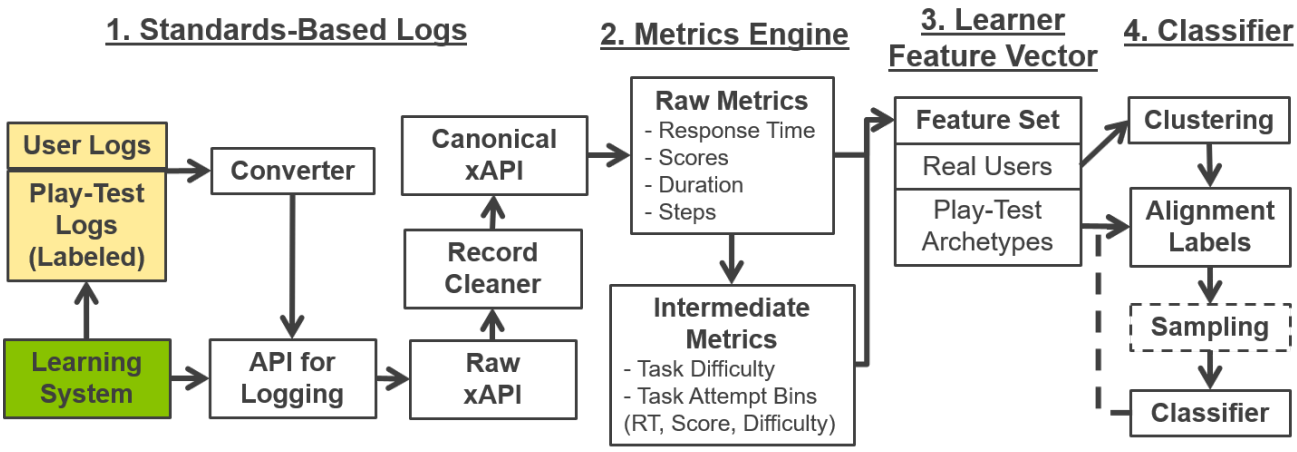


Figure 1: SMART-E Analytics Pipeline Phases

classifier is useful as opposed to simply using archetypes to label bottom-up clusters. Q4 and Q5 query the effectiveness of feature sets identified in the literature for classifying engagement, starting with a very minimal set (response time and scores) and then analyzing an expanded set of features for their impact on cold-start performance.

4. METHODS

To examine these questions, an analytics pipeline was developed and then applied to a data set from a scenario-based intelligent tutoring system. This section will briefly describe the pipeline, then the learning system that produced the data set, and finally the techniques used to investigate each research question.

4.1 Engagement Analytics Pipeline

While this paper focuses on a specific data set, the techniques applied here are designed to be generalizable and reusable as part of the SMART-E pipeline, shown in Fig. 1. This pipeline starts by standardizing the data available, recording data from an arbitrary learning system as learning records that meet the xAPI standard [1]. This "Raw xAPI" data may either be sent directly by the system (e.g., through an API for logging) or generated by running a converter on system logs after-the-fact. Raw xAPI data logs are then cleaned by a script (partially system-specific) which corrects common data problems, such as sessions that terminated improperly or missing data fields that can be inferred from other data. This ensures that the Canonical xAPI data store does not have missing data.

All xAPI records contain metadata which allow them to be structured into an activity tree, representing both sequential and parallel tasks. While the tree can be nested arbitrarily, four levels are analyzed to generate raw metrics tables: steps, tasks, lessons, and sessions. Raw metrics primarily record time-based information (e.g., duration of a task, response time for first step), score-based information (e.g., numerical score and/or correctness), and support used (e.g., hint counts, retrying a problem).

Metrics related to task skills are not calculated, since the majority of systems do not tag their tasks with a consistent ontology of knowledge components. Intermediate metrics are generated using feature construction calculations based on raw metrics, without analyzing the xAPI logs. For this work, the most important intermediate metrics are averages across attempts (e.g., average scores, average task duration), the average difficulty for each task (inferred from first-attempt scores), z-scores for task metrics (e.g., time-on-task for the learner relative to other users) and a Laplace-smoothed logarithm of each task duration (i.e., $ln(t+1)$). Additional metrics can be added fairly rapidly, if they rely on raw metrics.

Based on these metrics, feature vectors are generated that represent each learner's performance in the system. In the current work, these vectors rely on all of the learner task data for a session, though one could generate similar features for specific tasks, across multiple sessions, or for recent tasks in a session (i.e., any collection of tasks). First, two simple features were calculated: average response time across tasks (Avg. RT) and average task performance (Avg. Score). These were considered the minimal information to potentially infer engagement.

Next, a more complex feature set was developed to model interactions between task response time (RT), task scores, and task difficulty. Based on z-score cutoffs, the value of each variable was placed into one of three categories (low, medium or high) when possible, and into the most categories available when not (i.e., only medium if all values equal; only low and high if only two types of values). This was done based on a one-dimensional Gaussian distribution, with cutoff values at <33% (low), 33-66% (medium), and >66% (high). Further we ensured that each variable had at least 4 corresponding data-points in order to arrive at robust cutoffs (i.e., each unique task had been attempted by at least 4 different learners, to judge its difficulty, score and time distribution). Each scored task increments a bin associated with its three variables (e.g., RT=fast, score=high, difficulty=high will increment exactly one out of 27 possible bins). This binning approach is fairly general, and can be inferred using only standard logging data.

Since 27 bins will often be fairly sparse for an individual

learner, these were aggregated to form 7 bins which align to behavioral engagement patterns from the literature: Expert, Cherry Picking, Engagement/Diligent, Self-Regulated, Distracted, Racing, and Careless. These bins roughly correspond to the patterns we introduced earlier except we omitted Nominal Engagement, roughly equivalent to average, and we added a Careless bin focused on errors on easier tasks. The Expert bin was increased whenever high scores were obtained for difficult problems with only a normal or low delay or for high scores on ordinary problems done quickly. The Cherry Picking bin incremented for high scores with a low delay (regardless of problem difficulty). Engagement/Diligent was incremented when difficult problems were completed after a high delay. Self-Regulated was incremented when the amount of time spent on the problem was at least as high as the difficulty level, even if the score was not high. Distracted was triggered in the opposite case, where the time to respond was overly long for the difficulty of the problem. Racing was incremented for fast responses, either with low scores or with medium / high scores on easier problems. The Careless bin included only low scores on easy problems or low scores on medium difficulty problems when completing them quickly.

These bins were not mutually exclusive, since more than one behavior might explain a given interaction. Additionally, they are not validated and should be thought of as noisy constructs to bin low-level features, rather than necessarily predictive of their given labels. However, since these aggregation patterns are derived from the literature, these features are candidates that may be relevant across different systems, users, or data sets.

4.2 User Data: ELITE Scenarios

We use data from the system, ELITE Lite Counseling, designed for U.S. Army officers in training to learn leadership counseling skills, such as active listening, checking for underlying causes, and responding with a course of action [11]. Learners select what to say to virtual subordinates from a menu leading to different points in a branching graph representing the possible conversations. The virtual subordinates speak using pre-recorded audio and act via 3D animations.

Each learner choice can have both positive and negative annotations. Positive annotations correspond to correctly applying a skill such as active listening, and negative annotations correspond to omissions or misconceptions. Based on these annotations, a choice can be fully correct (only positive annotations) or two forms of incorrect: fully incorrect (only negative annotations) or mixed (both positive and negative annotations). For the pipeline, this was converted to two forms: a correctness category and a numerical score in which mixed answers were given partial credit (0.5) compared to correct answers (1.0) and incorrect answers (0.0).

Each simulated conversation is also followed by an After Action Review (AAR) in which learners are asked multiple-choice questions about all of their dialogue choices that were mixed or incorrect. For these AAR questions, if the first attempt to answer was successful the learner earned a score of 1; otherwise, the learner earned a score of 0 but had to keep trying until they selected the correct response.

The ELITE data set for this research included a corpus of 145 subjects from experiments described here [17] which we consider user data. Each "user" completed three scenarios: Scenario 1, Scenario 1 (Repeated), and Scenario 2. Due to the dialog trees, users did not all see the same decision tasks when completing the same scenarios. However, substantial overlap was observed for tasks and a majority of tasks were attempted by a significant number of users. For the purpose of estimating task difficulty, a threshold of 5 attempts was used, below which the difficulty and metrics relying on difficulty (e.g., binning) could not be calculated.

4.3 Play Persona Data

Play-testers were students and employees of the lab who volunteered their time, and generally were not familiar with the scenario content. For the ELITE system, only 5 play-test archetypes were reasonable to classify: Expert, Diligent, Nominal Engagement, Racing, and Distracted. Play-testers followed the same protocol (i.e., scenario 1, scenario 1 repeated, scenario 2) used to collect the user data. Thus, they had no direct control over the tasks they encountered, and so some patterns were unlikely to be observed (e.g., Self-Regulated and Cherry Picking).

Each play-tester was able to generate data for up to three archetypes, by attempting them in a specific sequence. First, they could play as either Diligent or Distracted. These roles could only be played at the beginning of testing to simulate a novice seeing the system for the first time. Next, a Racing run was completed; fast response times meant testers would still make errors despite their previous practice. Expert runs were collected in two ways: either an actual expert generated the data (2 sessions), or a tester carefully reviewed the correct answers (e.g., in the AAR) and/or was coached by an expert (13 sessions). These different methods for "expert" data produced similar results, though actual experts were slightly faster. In the unlabeled data, an archetype for Nominal Engagement was generated by extracting five clusters and assigning Nominal Engagement to the cluster not aligning with the other four archetypes.

Instructions for each play-test archetype were as follows. **Diligent:** spend as much time as you need on each choice to try to get the best answer, including reading carefully, and double-checking answers. **Distracted:** engage in one or more competing activities, including checking email and responding when relevant, browsing social media, engaging in a conversation, and eating. **Racing:** pretend you don't care much about the content, so you are doing the bare minimum and are fine with a so-so score to get done quickly. **Expert:** review content in-depth immediately ahead of time, and approach it with as many answers memorized or quickly-available as possible (e.g., in notes, from an expert) so you can answer well quickly. Of these, all except Distracted were easily understood by testers. Due to the lack of standardization for Distracted, some testers struggled to find a competing distraction task (e.g., did not use much social media, did not have high email volume, already ate lunch). In this case, a member of the research team asked the user questions or other requests to distract them. A total of 51 archetype sessions were collected, which may be more than necessary, since preliminary analyses found similar results with about 25 points balanced across classes.

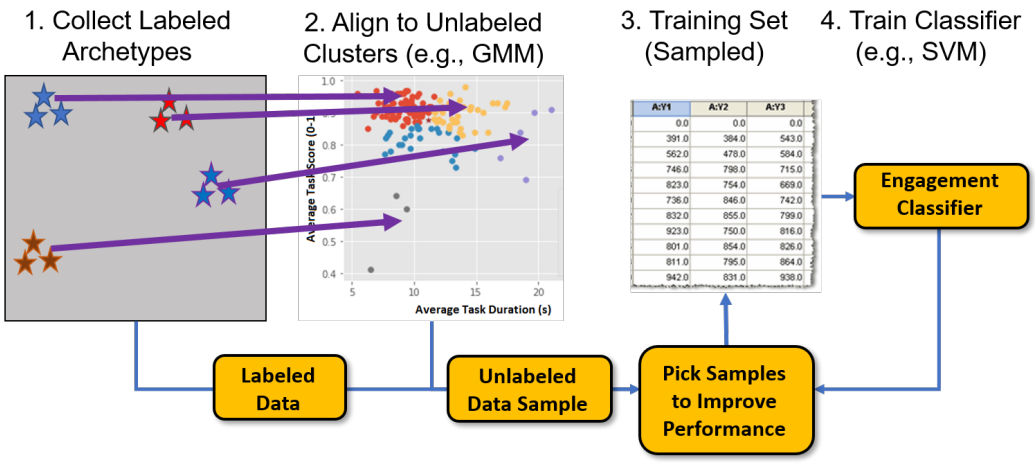


Figure 2: Semi-Supervised Classifier Training

4.4 Cluster Alignment Testing

As shown earlier in Fig. 1, both the real user data and the play-tester archetype data were processed by SMART-E to generate feature vectors that represent each individual. A number of techniques were then applied to generate labeled clusters. This cluster-labelling process allowed us to classify a learners' engagement coarsely on the basis of the cluster that they were assigned to.

First, user feature data was clustered bottom-up into five distinct clusters using k-means and Gaussian mixture models (GMM) methodologies as implemented in the scikit-learn package [30]. The number of clusters was verified through an elbow-curve analysis of variance explained (elbow at k=5). Exploration of k=4 and k=6 found both to be less stable; cluster assignments were often very different for subsamples of data points, with k=4 being particularly unstable.

For this analysis of the full sample, we associated each user cluster with a unique archetype (i.e., alignment of the smaller archetype clusters with the user clusters). The alignment was determined using the Hungarian Method (Kuhn-Munkres algorithm)[24], which is a global, optimal-matching algorithm which minimized the sum of the Euclidean distances between these user cluster centroids and archetype centroids. As noted previously, the Nominal cluster was determined as the cluster remaining after all archetype groups were matched. As a result, each of the user clusters (and consequently the points within that cluster) had an associated unique archetype which additionally served as its label. When this cluster alignment process is used as the only technique to label points, it will be referred to as *Clustering Alone*.

4.5 Semi-Supervised Classification

This technique of clustering alignment was compared against a semi-supervised approach that built a classifier using the play-test and user clusters. The high level concept of this semi-supervised classifier is shown in Fig. 2. The first two steps of the semi-supervised approach are the same as Clustering Alone. This generates a pool of weakly-labeled candidate labeled points. The points in this pool can be either taken as a full set to train a classifier model such as SVM or they can be sampled to incrementally train a classifier using active learning techniques until a stopping rule is hit (e.g., entropy sampling).

To compare the classifier against cluster-level labels based on archetype alignment alone, we calculated two quality metrics for the labels given to user sessions, which we will term *consistency* and *stickiness*. Consistency refers to the fraction of sessions that are labeled with the same engagement archetype which they would receive when the full data set is available. This is important because as data gets larger, unsupervised clusters are more likely to reflect the true distributions.

Stickiness refers to the likelihood that a user session retains the same engagement label after a batch of new data is added (similar to intra-rater reliability). This is important for actionable engagement metrics: if Student A is classified as Diligent, it will be confusing if Student B who completes an identical run is classified differently due to data that arrived in between. While this cannot be fully avoided, approaches that tend to keep the same label for an identical session will appear more fair and reliable, so that an instructor could be more confident in using the classifications.

That said, neither consistency or stickiness alone are sufficient for useful classification. For example, always assigning all users to the same category maximizes both metrics. However, assuming clusters for the full data set are reliable, then these measures help to identify how quickly and reliably labels approximate the final labels. This is important for addressing the cold start problem, so that engagement patterns can be quickly identified in a new system.

To calculate the number of samples to reach a given level of cold start performance, random splits were made of the user data set into train-test subsets (115 train, 30 test). For each random split, the classifier was trained using the archetype data set (51 samples) and increasingly larger subsets of the user training data in increments of 5. When evaluating cold-

start performance, a consistency of 85% was considered a reasonable target threshold for reliability against the full sample. While the actual consistency required will depend on the specific application, this cutoff should give some insight into how quickly different approaches converge toward their larger-sample performance.

Since the pipeline parameterizes the specific algorithms, follow-up exploratory analyses were conducted with different types of clustering algorithms (e.g., k-means, GMM), classification algorithms (e.g., logistic regression, support vector machines), and semi-supervised sampling algorithms (e.g., full sampling, margin sampling with stopping rules to exclude certain unsupervised samples). Different combinations of these algorithms did not show qualitatively different end-results on these metrics, and any differences were not conclusive (e.g., GMM clusters appeared slightly more stable than k-means as data was added, but within random variation). As a result, this paper presents results for the GMM clustering with a Support Vector Classifier, where these results are representative for the different approaches explored.

5. RESULTS

Focusing on GMM clustering, we revisit the alignment of the five user clusters with the play-tester archetype groups. The clusters were generated using the average of the logarithms for task response time (Log-RT) and average of task scores (Scores). Fig. 3 plots the real user data with unsupervised clusters. Table 1 shows feature means and standard deviations for each archetype, above its most closely-aligned bottom-up cluster. Note that while Log-RT was used for clustering, the actual time in seconds is given in the table and figure for easier interpretation.

Despite being generated independently, the play-test data closely resembles the real bottom-up clusters. As a trend, the play-tester archetypes tend to be more extreme (i.e., farther from the average user) than the clusters they align to. This is likely due to play-testers acting out more exaggerated or consistent patterns than real users. However, this may actually be an advantage, since play-test archetype data points may be more likely to be outliers in the vector space and good anchors for distinct clusters. The results from Table 1 support research question Q1, in that play-testers were able to act out similar patterns as real users and that the play-tester data showed fairly distinct groupings (as evident in the standard deviation values). One exception was the Distracted archetype, which had a very high variance for time compared to real users in the corresponding cluster. However, despite the high variance, the Distracted archetype data remained distinct from other archetypes' data.

5.1 Reliability vs. Expert Labels

The validity of this alignment on the full data set was evaluated by surveying a set of external engagement experts (N=5) to label the same bottom-up clusters obtained from the user feature data, based on the descriptions of the engagement archetypes. Selection criteria for experts required a Ph.D. in a relevant area, publishing at least one substantial paper researching learner engagement, and having no prior experience with the data set.

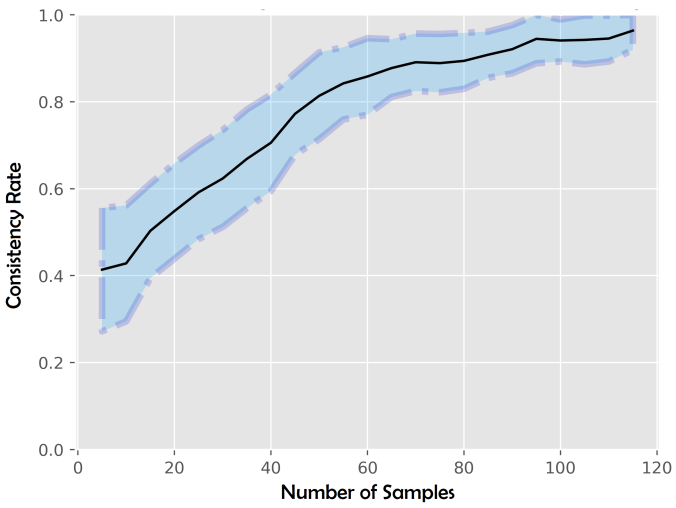
Experts labeled cluster graphs (e.g., Fig. 3) generated by both k-means and GMM, and maintained quite similar labels across each (76% agreement). Since the clusters and labels for both GMM and k-means were very similar, all labels were treated as examples from the same task. Inter-rater reliability metrics were moderate between experts: 55% Agreement; Fleiss' kappa = 0.44; Krippendorff's alpha = 0.45. Expert raters had very high reliability for Expert and Racing labels, but approximately half of experts demonstrated a consistently different interpretation for Diligent, Nominal (phrased as "Average" in the survey), and Distracted. Based on open response comments, this may have been the result of interpreting minor wording differences in the prompts (e.g., "novice learners" for Diligent vs. "learners" in Distracted).

The human labels for clusters were then compared pairwise against the automated alignments, resulting in Agreement, Fleiss' Kappa and Krippendorff's alpha metrics which were higher than within-experts though still in the moderate range: 66% Agreement; Fleiss' kappa = 0.57; Krippendorff's alpha = 0.58. Given these results and expert sensitivity to the wording of archetype descriptions, we conclude that the automatic alignment appears to be at least as useful as expert consensus ratings for labeling engagement clusters. We anticipate automatic alignment to be even more advantageous when the feature space expands beyond 3 dimensions, making it difficult for human experts to visualize or evaluate.

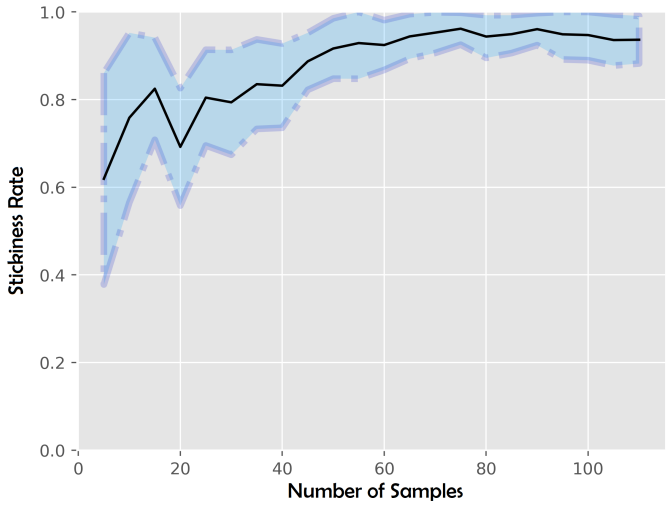
Group	N	Avg. RT (s)	Avg. Score
Expert (Arch)	15	8.53 ± 2.43	0.95 ± 0.04
Cluster 1	25	8.10 ± 1.00	0.93 ± 0.03
Diligent (Arch)	14	13.15 ± 3.83	0.89 ± 0.07
Cluster 2	75	11.06 ± 1.61	0.90 ± 0.03
Nominal (Arch)	-	-	-
Cluster 3	13	8.63 ± 1.11	0.82 ± 0.02
Distracted (Arch)	12	22.27 ± 13.80	0.77 ± 0.17
Cluster 4	28	15.81 ± 3.43	0.83 ± 0.07
Racing (Arch)	10	7.18 ± 2.47	0.56 ± 0.17
Cluster 5	4	7.98 ± 1.08	0.55 ± 0.09

Table 1: Cluster vs. Archetype Centers ($\mu \pm \sigma$)

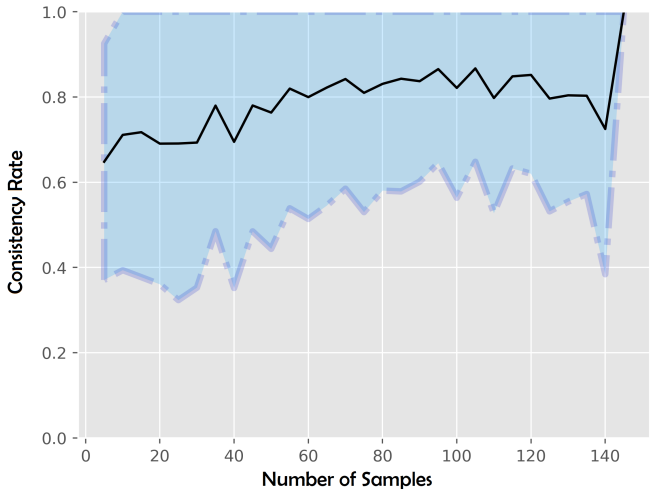
Figure 3: GMM User Clusters for Response Time and Score Features



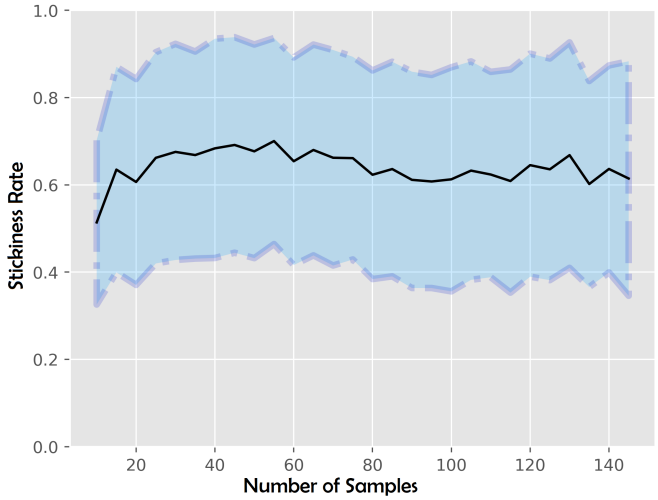
(a) Consistency: Semi-Supervised SVM



(b) Label Stickiness: Semi-Supervised SVM



(c) Consistency: Clustering Alone



(d) Label Stickiness: Clustering Alone

Figure 4: Consistency and Stickiness for Semi-Supervised vs. Clustering Alone

5.2 Consistency of Semi-Supervised vs. Clustering Alone

To evaluate how play-test data can be used to classify new user sessions, a semi-supervised approach was explored which trained a Support Vector Machine (SVM) classifier using both the play-test archetype data and the data from the bottom-up cluster that best aligned to each archetype, with test-set labels determined by the classifier. For the clustering alone comparison case, bottom-up clusters were directly aligned against archetype data to determine their labels and test-set labels were determined based on their closest cluster. 20 random splits were made of the user data set into train-test subsets (115 train, 30 test). For each random split, the classifier was trained using the archetype data set (51 samples) and increasingly larger subsets of the training data set in increments of 5, and then evaluated.

Consistency was calculated against the test set of 30 samples. Stickiness of labels was calculated for each set of labels against the prior set (e.g., model trained on N samples vs. N-5 samples). Due to the higher level of noise for clustering alignment alone, 100 runs were conducted instead of 20 for a smoother average. These results indicated that training a classifier which combined both types of data produced higher consistency and less variation. Specifically, on the basic features (avg. RT and performance only), the semi-supervised SVM reached 85% average consistency at 52 samples (Fig. 4a), while aligned clusters alone required 95 samples to reach this level (Fig. 4c). Clustering alone was more consistent with the full-data cluster labels until approximately 25 samples (i.e., when the user data reached approximately half of the archetype data).

Likewise, the stickiness of labels as data increased reached an average of 85% by 45 samples for the semi-supervised classifier (Fig. 4b). Clustering alone never reached 85% and remained less than 70% on average (Fig. 4d). For both metrics, the variance (blue bars) were larger for clustering alone. One reason for greater variability for clustering alone

is that sparse data for certain cluster regions (e.g., Racing, with only 4 real users), so alignment alone may try to align a non-existent cluster given limited data. However, the semi-supervised classifier appears to mitigate this issue since training is anchored by play-test data points.

These analyses were performed using both the basic features and the expanded feature set (e.g., bins that count instances of engagement behavior patterns based on response time, score, and difficulty categories). Both feature sets required a similar number of samples to reach the same level of consistency (e.g., about 85% consistency after 50 samples). While it is possible that the expanded feature set might produce more valid labels for an instructor (e.g., better reflecting the categories of users who an instructor might follow-up with), this will not be due to improved cold-start performance.

5.3 Semi-Supervised Class-wise Consistency

An analysis of the consistency for labels in individual clusters (Fig. 5) shows similar insights to the overall clustering label consistency. Points in larger clusters (e.g., Diligent, Expert) are consistent fairly quickly. However, small clusters (Racing) may have few/zero examples even when considering as many as 40 data points, and even with 100 data points have poor consistency. As such, classes with few examples might only be useful for a smaller set of use-cases (e.g., sufficient to share with an instructor, but possibly not reliable enough to take an automated action confidently). We also note that based upon the stickiness analysis (Fig. 4d), performance may be limited by the instability of clustering (points moving between clusters even with nearly full data).

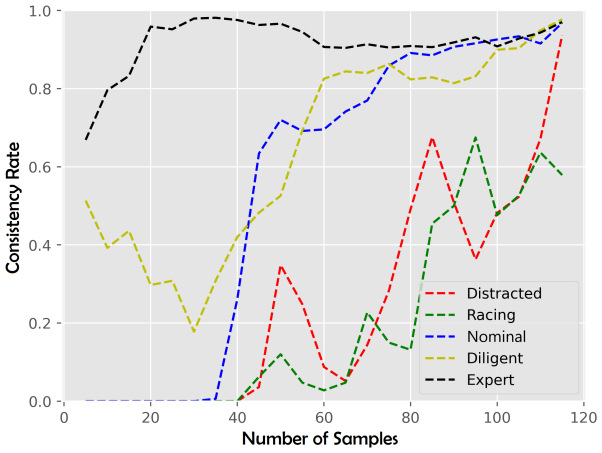


Figure 5: Class-wise Consistency for Semi-Supervised SVM

5.4 Semi-Supervised vs. Final Clusters

The semi-supervised results were compared for their agreement with the labels obtained via alignment with the final clusters generated using the full data set. This final-clusters reference point (see Fig. 3) was used to calculate average accuracy, precision, recall and F-scores (Fig. 6), as a function of the increasing dataset size. While final clusters are not a perfect reference, it shows that accuracy versus final clusters increases fairly rapidly, but that precision, recall and F-scores are consistently lower.

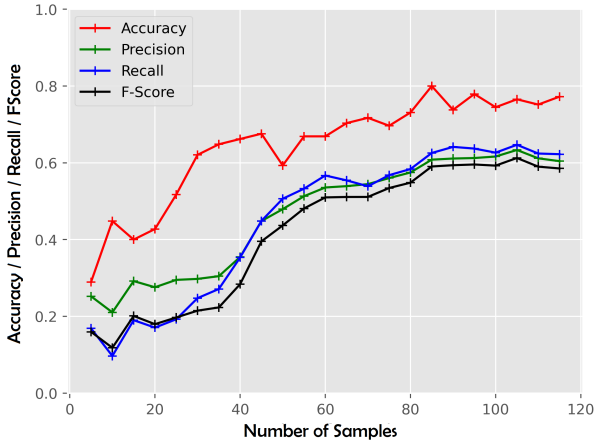


Figure 6: Agreement of Semi-Supervised SVM vs. Final Cluster Labels

6. DISCUSSION

Based on the results presented, this work demonstrates the feasibility of using a play-testing methodology for detecting behavioral patterns of engagement. Moreover, this work also found that a classifier could be developed using this approach without engineering application-specific features. The classifier also offered reasonable cold-start performance and labeled engagement data fairly consistently for 5 categories after 52 unlabeled samples and 51 archetype samples.

Of the five research questions investigated, there was positive support for four answers, with one left indeterminate. For Q1, play-tester data was distinctive and archetype data followed coherent patterns on features (e.g., response time, correctness). Archetype data did not show substantial overlap between archetypes, even though play-testers received only limited instructions. This may be due to the limited degrees of freedom for the task. In a more complex or open-ended system, increased variation might lead to less coherent archetype data. With that said, many systems have similar characteristics to the ELITE scenarios studied here (i.e., sequential linear or branching choice tasks, mixed with passive content such as videos or animations). Moreover, these kinds of systems are often problematic for engagement, such as mandatory corporate training modules.

For Q2, it was demonstrated that automated matching of play-test archetypes against pre-defined clusters performed comparably to expert labels for the same clusters. While refining instructions might improve inter-rater reliability on this specific task, the features presented to experts were already chosen to be simple and visualizable so this represented an optimistic scenario for expert cluster labeling. On more complex feature sets or systems, expert analysis might not even be possible. The broader question not explored in this work is the machine vs. human play-test agreement if they were not given pre-defined clusters (i.e., a data exploration task). However, this would be challenging to conduct: it requires a deep analysis by each expert researcher and the types of engagement categories might be highly uneven. Alternatively, archetypes might be determined from already-analyzed data sets (e.g., such as for hint-abuse), to

see how effectively traces of play-test disengagement might match authentic disengagement patterns.

For Q3, it was established that training a classifier with both play-test data and unsupervised cluster data showed advantages over simply re-clustering with new unsupervised data and then aligning clusters to archetype data. In some respects, this is not surprising: while the consistency metric used for evaluation is based on the unsupervised results from the full data set, the classifier is able to train with more data up-front (as much as double initially). More importantly, since all key archetypes are present in the play-test data, no category will start unrepresented. This particularly helps for classifying points from relatively rare but distinctive categories (e.g., Racing). However, despite this advantage, points in small classes remained substantially less consistent than those in larger classes.

As a long term issue, it is an open question about the best way to mix this data. Neither data source represents ground truth. The archetype data demonstrates coherent engagement patterns, but these patterns might not reflect the ways real users experience the system (e.g., in the current research, they were exaggerated/overly extreme). The real user data is authentic, but may slowly wash out the classifier with unremarkable samples (e.g., overly ordinary). Exploratory work was conducted where stopping rules were applied to balance the number of archetype vs. authentic samples (derived from active learning techniques, such as margin sampling and entropy sampling), but this has not yet produced obvious improvements. Similarly, techniques for weighting samples might be applied. However, the ideal balance between these data sources probably depends on the target use-case for the classifier. A recommender system may want a classifier that acts on labels regardless of their confidence scores. By comparison, a human instructor might prefer a narrowly-scoped but highly-actionable classifier, which might detect clear outliers but allow the majority of user sessions to be in a non-descript "Nominal" category or not confidently classified.

On questions about the features required to classify engagement, we found that basic features for the log of response times and scores were sufficient in this case (Q4) but did not show improvement with the expanded feature set including task difficulty and feature interactions improved classification (Q5). These features helped to detect engagement behavior that matched patterns observed from play-testing: Expert/Recall, Diligent, Racing, and Distracted as well as Nominal (i.e., matched by exclusion). However, both k-means and GMM tended to split up the mass of points in the region of Expert, Diligent, and Nominal despite these clusters being adjacent to each other. The cluster-alignment approach used in this work was selected primarily for the ability to interpret cold-start trends, while more advanced methods should further improve performance. It might be preferred to investigate techniques such as anomaly detection, which would favor a larger central cluster and smaller outliers which could correspond to atypical behavior which is actionable. Alternatively, alternate semi-supervised techniques are available, such as applying specialized semi-supervised support vector machines (which optimize margins for both labeled and unlabeled data) [8, 31] or more advanced techniques for integrating cluster data [16]. While expanded features did not improve consistency or stickiness metrics (Q5), other systems may still benefit from expanded features. However, additional features also increase the required data and may result in overfitting, need attenuating/filtering features during clustering, or other trade-offs. As such, further research is needed on this problem.

7. CONCLUSIONS AND FUTURE WORK

Based on these findings, this work contributes a number of novel approaches to analyzing engagement. First, this research demonstrates the utility of play persona data gathered during professional or quality assurance testing for training useful data mining algorithms. Since there is no definitive metric for engagement, play-test data offers an additional distinct data source to help recognize engagement and disengagement. To our knowledge, this approach has not been applied to analyzing engagement in learning.

Second, this approach offers advantages over current approaches for cold-start labels. Since the behavioral intentions of the play-test users is known with confidence, these labels offer a good data set to help overcome cold start problems. As compared to traditional approaches such as training observers or collecting in-the-moment self-reported engagement [13, 29], play persona data can be collected prior to real system users. This approach also allows balanced sampling for important but lower-frequency engagement behaviors (such as racing, in this analysis).

Third, we have demonstrated that semi-supervised classifiers trained based on a combination of play-test labels and unlabeled data offer more consistent labels than relying on clustering alone, which has been used to analyze engagement behaviors [23]. Moreover, as shown by agreement with expert labels at the cluster level, the alignment approach can provide similar insights without manually interpreting clusters. While expert interpretation is still ideal, this allows immediate insights without waiting for an expert analysis.

This approach is also pragmatic: System developers should already test and perform quality assurance on their software and content [35]. Behavioral archetype data can be collected during this process, by having testers play out engagement styles in a prescribed order based on their expected learning. Moreover, this work is not unique to specific archetypes: if learners are expected to engage in different patterns, play-testers may be able to produce those patterns instead. However, not all archetypes may be realistically playable by testers. For example, experts cannot typically generate novice answers. As such, this approach may be most effective when testers are similar to authentic users. As such, future work will explore how expert observer labels and self-report data might complement this play persona data.

8. ACKNOWLEDGMENTS

This research was sponsored by U.S. Army through the USC ICT University Affiliated Research Center (W911NF-14D-0005). However, all statements in this work are the work of the authors alone and do not necessarily reflect the views of sponsors, and no official endorsement should be inferred.

9. REFERENCES

[1] Advanced Distributed Learning. *xAPI Specification*, 2020.

[2] R. D. Axelson and A. Flick. Defining student engagement. *Change: The magazine of higher learning*, 43(1):38–43, 2010.

[3] R. S. Baker, A. T. Corbett, K. R. Koedinger, S. Evenson, I. Roll, A. Z. Wagner, M. Naim, J. Raspat, D. J. Baker, and J. E. Beck. Adapting to when students game an intelligent tutoring system. In *International Conference on Intelligent tutoring systems (ITS)*, pages 392–401. Springer, 2006.

[4] R. S. Baker, S. K. D'Mello, M. M. T. Rodrigo, and A. C. Graesser. Better to be frustrated than bored: The incidence, persistence, and impact of learners' cognitive-affective states during interactions with three different computer-based learning environments. *International Journal of Human-Computer Studies*, 68(4):223–241, 2010.

[5] R. S. Baker and L. M. Rossi. Assessing the disengaged behaviors of learners. *Design recommendations for intelligent tutoring systems*, 1:153–163, 2013.

[6] J. E. Beck. Engagement tracing: using response times to model student disengagement. In *International Conference on Artificial intelligence in Education (AIED)*, pages 88–95. IOS Press, 2005.

[7] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[8] O. Chapelle, V. Sindhwani, and S. S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9(Feb):203–233, 2008.

[9] S. L. Christenson, A. L. Reschly, and C. Wylie, editors. *Handbook of Research on Student Engagement*. Springer, New York, 2012.

[10] M. Cocea and S. Weibelzahl. Disengagement detection in online learning: Validation studies and perspectives. *IEEE transactions on learning technologies*, 4(2):114–124, 2010.

[11] M. G. Core, K. Georgila, B. D. Nye, D. Auerbach, Z. F. Liu, and R. DiNinni. Learning, adaptive support, student traits, and engagement in scenario-based learning. In *Proc. of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, 2016.

[12] R. S. d Baker, A. T. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International conference on intelligent tutoring systems*, pages 406–415. Springer, 2008.

[13] M. A. A. Dewan, M. Murshed, and F. Lin. Engagement detection in online learning: a review. *Smart Learning Environments*, 6(1):1, 2019.

[14] S. D'Mello and A. Graesser. Dynamics of affective states during complex learning. *Learning and Instruction*, 22(2):145–157, 2012.

[15] S. D'Mello, A. Olney, C. Williams, and P. Hays. Gaze tutor: A gaze-reactive intelligent tutoring system. *International Journal of Human-Computer Studies*, 70(5):377–398, 2012.

[16] H. Gan, N. Sang, R. Huang, X. Tong, and Z. Dan. Using clustering analysis to improve semi-supervised classification. *Neurocomputing*, 101:290–298, 2013.

[17] K. Georgila, M. G. Core, B. D. Nye, S. Karumbaiah, D. Auerbach, and M. Ram. Using Reinforcement Learning to Optimize the Policies of an Intelligent Tutoring System for Interpersonal Skills Training. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019.

[18] Y. Gong and J. E. Beck. Towards detecting wheel-spinning: Future failure in mastery learning. In *Proceedings of the second (2015) ACM conference on learning@ scale*, pages 67–74, 2015.

[19] J. F. Grafsgaard, J. B. Wiggins, A. K. Vail, K. E. Boyer, E. N. Wiebe, and J. C. Lester. The additive value of multimodal features for predicting engagement, frustration, and learning during tutoring. In *International Conference on Multimodal Interaction (ICMI)*, pages 42–49. ACM, 2014.

[20] R. K. Hambleton, H. Swaminathan, and H. J. Rogers. *Fundamentals of item response theory*. Sage, 1991.

[21] C. Holmgård, A. Liapis, J. Togelius, and G. N. Yannakakis. Evolving models of player decision making: Personas versus clones. *Entertainment Computing*, 16:95–104, 2016.

[22] R. Janning, C. Schatten, and L. Schmidt-Thieme. Perceived task-difficulty recognition from log-file information for the use in adaptive intelligent tutoring systems. *International Journal of Artificial Intelligence in Education*, 26(3):855–876, 2016.

[23] M. Khalil and M. Ebner. Clustering patterns of engagement in massive open online courses (moocs): the use of learning analytics to reveal student categories. *Journal of computing in higher education*, 29(1):114–132, 2017.

[24] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[25] B. Lehman, S. K. D'Mello, A. C. Strain, M. Gross, A. Dobbins, P. Wallace, K. Millis, and A. C. Graesser. Inducing and tracking confusion with contradictions during critical thinking and scientific reasoning. In *International Conference on Artificial Intelligence in Education (AIED)*, pages 171–178, 2011.

[26] D. J. Leiner. Too fast, too straight, too weird: Post hoc identification of meaningless data in internet surveys. *SSRN Electronic Journal*, 2013.

[27] E. Mattheiss, M. Kickmeier-Rust, C. Steiner, and D. Albert. Approaches to detect discouraged learners: Assessment of motivation in educational computer games. *Proceedings of eLearning Baltics (eLBa)*, 10:1–10, 2010.

[28] B. D. Nye, S. Karumbaiah, S. T. Tokel, M. G. Core, G. Stratou, D. Auerbach, and K. Georgila. Engaging with the scenario: Affect and facial patterns from a scenario-based intelligent tutoring system. In *International Conference on Artificial Intelligence in Education*, pages 352–366. Springer, 2018.

[29] J. Ocumpaugh. Baker rodrigo ocumpaugh monitoring protocol (bromp) 2.0 technical and training manual. *New York, NY and Manila, Philippines: Teachers*

College, Columbia University and Ateneo Laboratory for the Learning Sciences, 60, 2015.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[31] T. Sakai, M. C. du Plessis, G. Niu, and M. Sugiyama. Semi-supervised classification based on classification from positive and unlabeled data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2998–3006. JMLR. org, 2017.

[32] A. Tychsen and A. Canossa. Defining personas in games using metrics. In *Proceedings of the 2008 Conference on Future Play*, 2008.

[33] S. C. Weissgerber, M.-A. Reinhard, and S. Schindler. Study harder? the relationship of achievement goals to attitudes and self-reported use of desirable difficulties in self-regulated learning. *Journal of Psychological and Educational Research*, 24(1):42, 2016.

[34] F. Wiltgren. 8 archetypes for break-testing your game, 2015.

[35] B. M. Winn. The design, play, and experience framework. In *Handbook of research on effective electronic gaming in education*, pages 1010–1024. IGI Global, 2009.

[36] N. Yee. The gamer motivation profile: What we learned from 250,000 gamers. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, pages 2–2, 2016.

Learning student program embeddings using abstract execution traces

Guillaume Cleuziou
University of Orléans, INSA Centre Val de Loire,
LIFO EA 4022
Orléans, France
guillaume.cleuziou@univ-orleans.fr

Frédéric Flouvat
University of New Caledonia, ISEA EA 7484
Nouméa, New Caledonia
frederic.flouvat@unc.nc

ABSTRACT

Improving the pedagogical effectiveness of programming training platforms is a hot topic that requires the construction of fine and exploitable representations of learners' programs. This article presents a new approach for learning program embeddings. Starting from the hypothesis that the function of a program, but also its "style", can be captured by analyzing its execution traces, the *code*2*aes*2*vec* method proceeds in two steps. A first step generates abstract execution sequences (AES) from both predefined test cases and abstract syntax trees (AST) of the submitted programs. The *doc*2*vec* method is then used to learn condensed vector representations (embeddings) of the programs from these AESs. Experiments performed on real data sets shows that the embeddings generated by *code*2*aes*2*vec* efficiently capture both the semantics and the style of the programs. Finally, we show the relevance of the program embeddings thus generated on the task of automatic feedback propagation as a proof of concept.

Keywords

Representation Learning, Program Embeddings, Neural Networks, Educational Data Mining, Computer Science Education, doc2vec.

1. INTRODUCTION

Increasingly, programming is being learned through the use of online training platforms. Typically, learners submit their code(s) and the platform returns any syntax errors or functional errors (typically based on test cases defined by the teacher). The exploitation of these data opens up new perspectives to monitor and help beginners in learning programming. They can be used, for example, to identify students who are dropping out, to target bad practices or to propagate teacher feedbacks. These functionalities would allow the learner to be more autonomous during his learning, and the teacher to be more reactive and efficient in his interventions. However, this exploitation requires a detailed analysis of the submitted programs. These training platforms must go beyond a simple syntactic analysis of the script, and allow the associated semantics to be considered. For this purpose, learning program embedding has recently emerged as a promising area of research [15, 13, 17, 7, 2, 3]. A natural way to generate such vectorial and condensed representations is to consider a computer program as a text and to exploit methodologies inspired by Text Mining.

Text mining has attracted a lot of interest in recent years. The representation of texts as vectors of real numbers, also called "embedding", has been at the heart of many recent works. These representations make it possible to project (or 'embed') a whole vocabulary into a low-dimensional space. Moreover, such a representation of words allows to exploit a wide variety of numerical processing methods (neural networks, SVM, clustering, etc.). At that stage, one of the challenges is to capture in these representations the underlying semantic relationships (e.g. similarities, analogies). The work of [11] based on the use of neural networks has been a precursor in this area. Their *word*2*vec* method is one of the most referenced in the field. Its principle is based on the relation between a word and its context (words appearing before and after). To do this, they propose some simple and efficient architectures to learn word embeddings from a corpus of texts. For example, the *CBOW* (Continuous Bag-Of-Words) architecture trains a neural network to predict each word in a text given its context. Their results show the ability of this approach to extract complex semantic relations (analogies) from simple operations on $v()$ projections, s.t. $v(\text{"}king\text{"}) - v(\text{"}man\text{"}) + v(\text{"}woman\text{"}) \approx v(\text{"}queen\text{"})$ or $v(\text{"}Paris\text{"}) - v(\text{"}France\text{"}) + v(\text{"}Italy\text{"}) \approx v(\text{"}Rome\text{"})$.

The transposition of these approaches to computer programs is not straightforward. The code has certain specificities that need to be integrated to have such rich representations [1]. Unlike texts, codes are runnable, and small modifications can have significant impacts on their executions. A program can also call other programs that can themselves call other programs. The context in which an instruction is used is also particularly important in deducing its role. Finally, unlike texts, program syntax trees are usually deeper and composed of repeating substructures (loops). Existing approaches for building program embeddings only partially integrate these specificities. They independently exploit the instructions [3], the inputs/outputs [15], part of the execution traces [17] or the abstract syntax tree (AST) [2]. They

Guillaume Cleuziou and Frédéric Flouvat "Learning student program embeddings using abstract execution traces". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 252-262. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

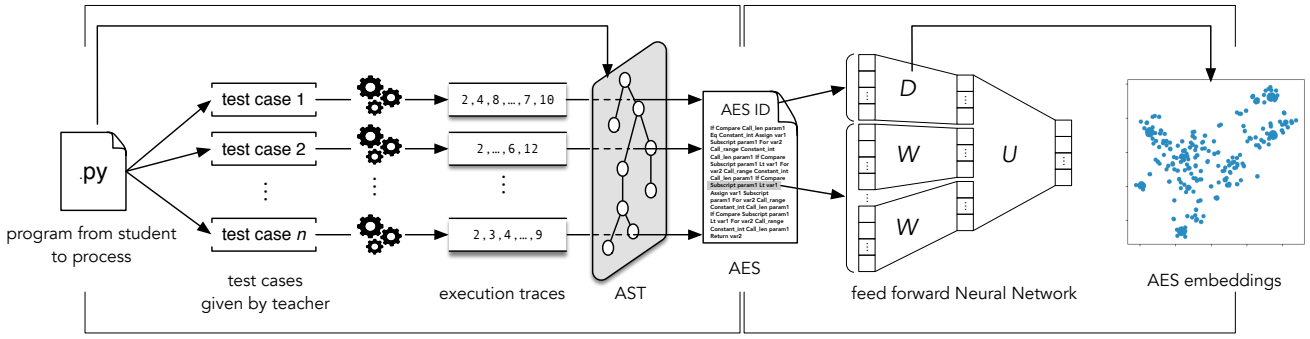


Figure 1: General scheme of the *code2aes2vec* method for learning program embeddings.

focus more on the function of the program (what it does) than on its style (how it does). Moreover, most of these approaches are supervised and build embeddings for a specific task (e.g. predict errors, predict functionality, etc.).

In view of these limitations, we propose the *code2aes2vec* method, exploiting instructions, code structure and execution traces of programs, in order to build finer program embeddings. Figure 1 schematizes the overall approach *code2aes2vec* we propose. The first step of this method consists in generating Abstract Execution Sequences (AES) from traces obtained on test cases executions and program ASTs. The second step uses the *doc2vec*[1] neural network [8] to learn program embeddings from AESs. Contrary to existing approaches, we therefore propose a generic and unsupervised method that learns program embeddings by using functional, stylistic and execution elements. This aspect is crucial in our application to be able to differentiate programs answering to the same exercise (i.e. implementing the same functionality) but in different ways (in terms of strategy or efficiency). Our approach is validated on two real data sets, composed of several thousands of Python programs from educational platforms. On these datasets, we show that embeddings generated with *code2aes2vec* allow to efficiently detect the function and the style of a program. In addition, we present a proof of concept of the use of such embeddings to propagate teacher feedbacks.

To summarize, the main contributions of this paper are :

1. the definition of a new (intermediate) program representation, called Abstract Execution Sequences (AES), allowing to capture more semantics,
2. exploiting these (intermediate) representations with *doc2vec* to build program embeddings in an unsupervised way,
3. the diffusion to the community of two enhanced datasets from educational platforms in computer science,
4. a proof of concept on the use of such program embeddings for feedback propagation for educational purposes.

The next section details existing works in the field and the originality of our approach in relation to it. Section 3 presents our two-steps method: the construction of abstract execution sequences (AES) and the learning of embeddings from them. Section 4 is devoted to the qualitative and quantitative evaluations of the learned representations before drawing up the many perspectives of this work (section 5).

2. RELATED WORKS

Learning representations from programs is at the heart of many recent works. They aim to embed this data in a semantic space from which further analysis can be conducted, because the generated representations (vectors of real values) are directly exploitable by a large part of the learning algorithms. To do this, recent work relies heavily on methods developed to build word embeddings in texts, while trying to integrate the specificities of the code.

These program embeddings are then used for prediction or analysis tasks related to the software development (debugging, API discovery, etc.) and teaching (learning programming). Two types of embeddings are more particularly studied: embeddings of the elements composing a program (words, tokens, instructions, or function calls) [12, 6, 7] and embeddings of the programs themselves [15, 17, 3, 4].

Nguyen et al. [12] study API call sequences to derive an API embedding that is independent of the programming language (thus allowing translation from one language to another). This embedding is learned by *word2vec* [11] from call sequences from several million methods. This method allows to build a word embedding from words appearing nearby in each text (i.e. its context). Two datasets derived from the Java JDK and from more than 7000 recognized C# projects from GitHub (more than 10 stars) are used for learning.

De Freez et al. [6] have a close objective, namely to build an embedding of functions used in a code in order to find

[1]a method derived from *word2vec* that allows to learn a document embedding from its words.

synonymous functions. However, the function calls made in each program are extracted in the form of a graph depending on the control structures. Random walks are then performed in this graph for each function, and the extracted paths are used as input to *word2vec* [11] to derive an embedding of functions. An extract of two million lines of the Linux kernel is used to learn the embedding.

In [7], the authors propose a relatively similar approach but replace function calls with abstract instructions. Each instruction sequence represents a path in the code depending on conditional instructions. In this way, it is similar to a trace even if the code is never executed. All possible paths are extracted but loop repetitions are ignored and only the most frequent instructions are considered (a threshold of 1000 occurrences is used in the experiments). Moreover, only certain constant values are considered to limit the size of the vocabulary. An embedding of these program instructions is then learned by the GloVe method [14] based on the co-occurrence matrix of the words. The authors also use a corpus of 311,670 procedures (in C language) from the Linux kernel, and evaluate it on a data set of 19,000 pre-identified analogies.

In many applications, it is not just a matter of considering the program's components but the program as a whole. Recent work has therefore focused on the construction of program embeddings.

Following a 'teaching' motivation, Piech et al. [15] construct student program embeddings and use them to automatically propagate teacher feedbacks (using the *k-means* algorithm). The embedding space is built from a neural network trained to predict the output of a program using its input. It thus captures the *functional* aspect of the code. The authors also try to capture the *style* of programs, using a recursive neural network based on the program's AST. Contrary to other approaches, the generated embeddings are matrices, not vectors, thus limiting their exploitation as inputs for next data analyses. They also don't consider learner-defined variables. The obtained representations actually capture quite well the code *function*, but fail in capturing the *style* of codes. The analyzed programs (from the Hour of Code site and a course at Stanford) are written in a language similar to Scratch and allow operations in a labyrinthine world.

In [17], the authors highlight the limitations of syntax-based approaches to capture the semantics of a program. Instead, they propose to consider the trace resulting from the code execution, and more precisely the values of the most frequent variables. Different representations are proposed and used to train a recurrent neural network whose objective is to predict errors made by students in a programming course. The embeddings of the programs corresponds to one of the neural network's layers. The authors put forward one representation more particularly, considering the trace of each variable independently and integrating the dependencies between variables in the structure of the neural network. However, the obtained embeddings are specific to one task, and this method requires to redefine the neural network architecture, with re-training, for each exercise.

Finally, Alon et al. [2] propose a neural network to predict the name of a method (i.e. the functionality) from its code. To do this, the program is first decomposed into a collection of paths (from one leaf to another) in the AST. Only the most frequent paths in the dataset are used as features (size constraints are also integrated). Then, the network learns which one is important for predicting the method name using the attention principle. The parameters of the trained neural network correspond partly to the final embeddings and partly to the weights supposed to quantify the importance of each (feature) path for the prediction task (attention principle). Training is performed on a corpus of more than 13 million Java programs from GitHub's 10,072 most popular projects. As mentioned by the authors, this approach requires a large number of input programs. Furthermore, it is not possible to predict the function (and embedding) of a program whose paths do not appear in the training set. The embeddings produced capture information and semantic relationships about the function of the code, but ignore style variants. Thus, two programs with the same function will be similar, regardless of how they have been coded. The quality of the analyzed code also has an impact on learning. The names given to the variables are particularly important for prediction.

3. THE *code2aes2vec* METHOD

Two main strategies emerge for learning program embeddings : by observing the results of program execution [15, 17] or by analyzing the script [3] and/or its AST [2]. Our approach is at the intersection of these two strategies and thus aims to take advantage of the functional and syntactic descriptions of the programs to induce relevant embeddings. We thus propose the *code2aes2vec* method which proceeds in two steps :

1. the *code2aes* step represents a program as an Abstract Execution Sequence (AES), corresponding to the AST paths used by the program during its execution on predefined test cases;

2. the *aes2vec* step uses a neural network to construct the embedding of the programs based on their AES (using the *doc2vec* approach [8]).

3.1 *code2aes*: construction of Abstract Execution Sequences (AES)

Translating a program into an AES requires providing, in addition to the program itself, a collection of test cases on which the program will be run in order to exploit its traces. In our educational context, the preparation of such a collection of test cases is not an additional effort since test cases are generally integrated into training platforms to evaluate submitted contributions. Moreover, this approach offers teachers the possibility of introducing verification choices and thus to drive the interpretation of his/her learners' programs according to his/her own pedagogical choices. For example, let's consider an exercise whose objective is to find a value in a table/list. A teacher wishing to emphasize algorithmic efficiency may choose to integrate a few unit tests for which the desired value appears early in the table. In such case, an efficient program stops the loop as soon as the desired value appears. These test cases will thus make it

possible to distinguish two (valid) programs based on their execution trace.

In practice the number of test cases provided by the teacher is quite small. We generally observe that less than ten test cases are enough to evaluate whether a program is correct or not.

Figure 2 illustrates the process of translating a program into an AES. This example considers as input the code submitted by a learner in response to the exercise "*write a Python function that returns the minimum value in an input list*". First, the AST is constructed. It describes the syntactic structure of the program in terms of control structures (`if-else`, `for`, `while`), function calls (`call`), assignments (`assign`), etc. Second, the code is executed on an example (here the input `[12, 1, 25]`) and its execution trace is kept, indicating the program lines successively executed. Finally, the AES is constructed by mapping these two levels of information: syntactic and functional. The sequence resulting from the trace is translated into a sequence of "words" extracted from the nodes of the AST.

Three levels of translation (or abstraction) are proposed according to the depth considered in the AST:

- AES level 0 : each program line is represented by a single word corresponding to the head symbol of the associated sub-tree in the AST (in red in Figure 2),
- AES level 1 : each program line is represented by one or more words corresponding to the head symbols of the associated sub-tree and its main sub-trees (in red and blue in Figure 2),
- AES level 2 : each program line is translated in a sequence of words corresponding to all the nodes appearing in the associated sub-tree (in red, blue and black in Figure 2).

For the last two levels, the names of the variables and parameters, as well as the values of the constants, have been normalized so as not to artificially extend the considered "vocabulary". Thus, the variable `res` is renamed `var1` and the variable `i` is renamed `var2`.

Note that each execution of a program on one test case generates a partial AES. A program will finally be represented by the concatenation of the partial AES obtained on each test case. An AES can thus be considered as a representative text of the program. Each partial AES corresponds to a sentence of this text.

3.2 *aes2vec* : learning program embeddings from AES

The *word2vec* method [11] is based on the distributional hypothesis of words in natural language [16] : a word can be inferred from its context. For example, the CBOW (Continuous Bag Of Words) version of *word2vec* allows to train an feed-forward Neural Network to predict a (central) word from its context. In *word2vec* a *context* is defined by the preceding and following words in the text. The structure of the neural network is reduced to a single hidden layer (encoding) ; the matrix W of the weights connecting the input layer to the hidden layer contains the word embeddings.

word2vec has already been used to learn token embeddings from a computer program [6, 3]. However the distributional hypothesis seems less satisfied on the tokens from a program than on the natural language, in particular because of the very limited size of the vocabulary and especially a little constrained compositionality (almost all combinations are observed).

[8] have proposed a variant of *word2vec*, aiming to learn simultaneously the embeddings of the words and the documents from which they are extracted. The *doc2vec* method is still based on the distributional hypothesis allowing to predict a word knowing its context, but this time the context integrates (in addition to the preceding and following words) the identifier of the document from which the word sequence comes from. In doing so, the authors introduce the idea that there are document specific variations in the natural/universal distribution of words in the language.

We exploit precisely this hypothesis of *document-based distributional variations* for the processing of AES built from the programs. We consider that each program, during its execution, generates different sequences of tokens (AES). We then use the DM (Distibutive Memory) version of the *doc2vec* algorithm to train a feed-forward Neural Network (with one hidden layer) to maximize the following log probability :

$$\mathcal{L} = \sum_{s=1}^{S} \sum_{i=k}^{T_s - k} \log p(w_i^s | w_{i-k}^s, \ldots, w_{i+k}^s, d_s) \qquad (1)$$

with S the total number of documents (or AESs), T_s the total number of words (or tokens) in document s, d_s the s^{th} document, w_i^s the i^{th} word in document d_s and k the size of the context on either side of the target word.

Figure 3 presents the architecture of the neural network used for *doc2vec* as we use it for learning program embeddings via their AES. The forward pass consists in first calculating the values of the hidden layer by aggregating the encodings of each word of the context and of the document : $h(w_{i-k}^s, \ldots, w_{i+k}^s; W, D)$ where $h()$ denotes an aggregation function to be defined (typically a sum, average or concatenation), W and D denoting the word embedding matrix (weight matrix between word inputs and hidden layer) and the document embedding matrix (weight matrix between document input and hidden layer) respectively. The output of the neural network can be interpreted as a probability distribution on the words of the vocabulary by applying an activation function (*softmax*) on the output $y_{w_i^s} = b + Uh(w_{i-k}^s, \ldots, w_{i+k}^s; W, D)$ where b is a bias term and U the weight matrix between hidden and output layer:

$$p(w_i^s | w_{i-k}^s, \ldots, w_{i+k}^s, d_s) = \frac{e^{y_{w_i^s}}}{\sum_j e^{y_j}} \qquad (2)$$

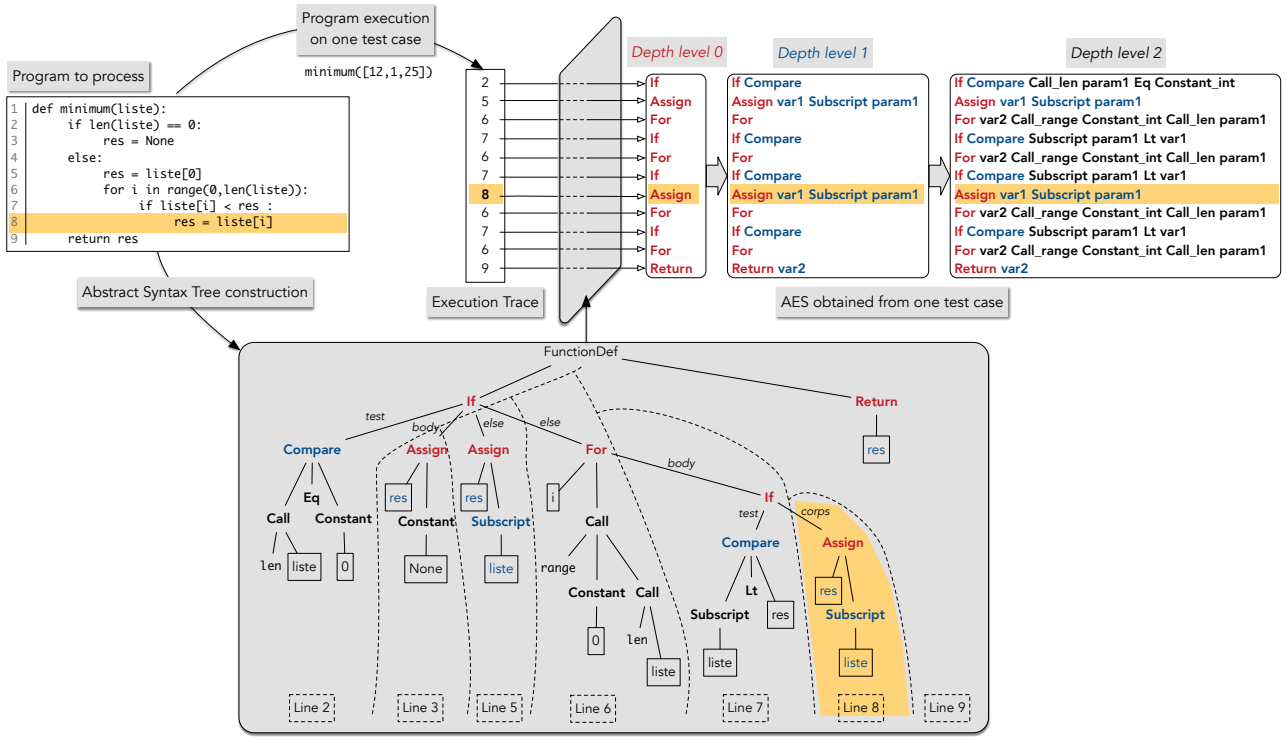


Figure 2: Construction of an AES from a Python program and a given test case.

Finally, the network weights are updated by stochastic gradient descent on the error, defined by the difference between the obtained output and the one-hot vector encoding the target word.

We consider programs as textual documents whose word sequence is given by an AES obtained by the previous step (*code2aes*). Indeed, as for text, the choice and the order of the "words" in an AES capture the semantic of the program, i.e. what the program does (its function) and how it operates (its style).

In this learning model, each AES vector (in D) is used only for predictions of the tokens from this AES, while token vectors (in W) are common to all AESs. The size of the vectors (for AES and tokens) is fixed and actually corresponds to the size of the desired representation space (embeddings).

Once the model is trained, the D matrix contains the embeddings of the programs. The positioning of a new program in this embedding space consists in inferring[2] a new column vector in D using the tokens from the new AES. The other parameters of the model remaining fixed (W as well as the *softmax* parameters).

Finally, let us mention that the choice of the aggregation strategy used in the hidden layer can be decisive. Indeed, a *sum* or *average* will consider each context as a bag-of-words (without taking into account the order), whereas a *concatenation* strategy offers the opportunity to exploit the order of words within the context. If the sequentiality (inside the context) is not a determining factor in the construction of embeddings for natural language, we will confirm in future experiments that the order of tokens is of high importance for learning program embeddings using AESs.

[2]Inference is made by purshasing the learning on the Neural Network.

4. EVALUATION OF THE APPROACH

4.1 Dataset presentation

Educational data are complex since programs may contain errors, be small in size, may not fully meet the intended functions and may be relatively redundant. These data have very different characteristics from the datasets used in software development. For our experiments, we thus built and use several real educational datasets (see Table 1). They consist of Python programs submitted by students on two training platforms in introductory programming courses. In addition to our (documented) *code2aes2vec* code for learning program embeddings, we also make available[3] these three "corpora" of Python programs, the associated test cases, and the AESs built on each program. All of the results presented in the rest of this section can thus be fully and easily reproduced.

The **NewCaledonia-5690** dataset (or NC-5690) includes the programs created in 2020 by a group of 60 students from the University of New-Caledonia, on a programming training platform[4]. The **NewCaledonia-1014** dataset (or NC-1014)

[3]*https://github.com/GCleuziou/code2aes2vec.git*

[4]Platform developed and made available by the CS department of the Orléans University Institute of Technology.

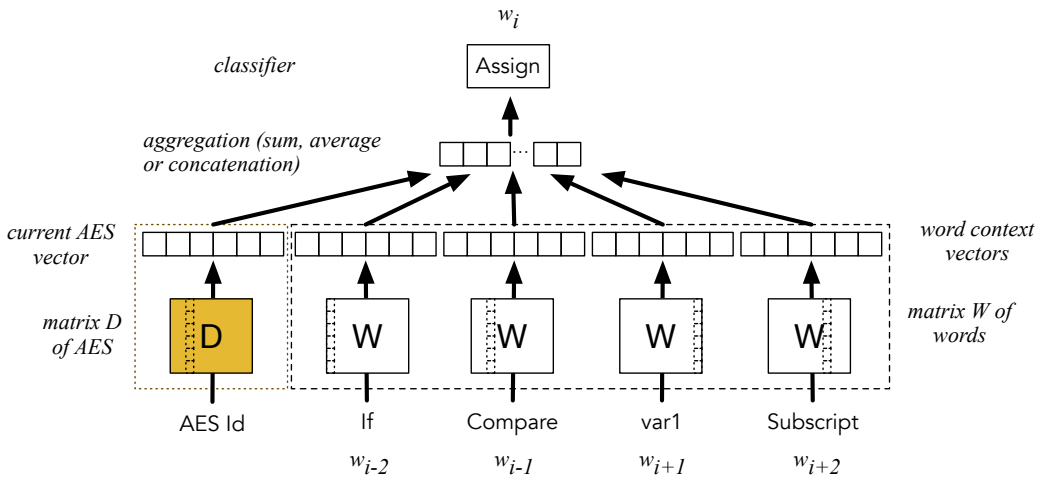


Figure 3: The neural network $aes2vec$ used to predict a word w_i from its original AES identifier and its context (here, two previous and two following words).

Table 1: Characteristics of the three Python datasets collected on programming training platforms. The 'Nb. of words' reported is the total size of the AES corpus; the number in parentheses indicates the size of the 'vocabulary'.

Datasets	NC-1014	NC-5690	Dublin-42487
Nb. programs	1,014	5,690	42,487
Avg nb. test cases per program	13.1	10.4	3.7
Nb. correct programs	189	1,304	19,961
Nb. exercises	8	66	65
Nb. 'words' AES-0	113,223 (20)	761,726 (44)	7,4 M (38)
Nb. 'words' AES-1	226,682 (42)	1,7 M (57)	15,2 M (83)
Nb. 'words' AES-2	690,019 (71)	3,9 M (113)	40,4 M (209)

Table 2: Exercises in the NewCaledonia-1014 dataset.

Exercice	Statement	#test cases
swapping	swap items in a list	4
minimum	look for the minimum in a list	8
compareStrings	compare two strings	11
fourMore100	return the first four values greater than 100 from an input list	7
indexOccurrence	return the index of the first occurrence of an item in a list	7
compareDates	compare two dates from their day, month and year	30
polynomial	return the roots of a polynomial of degree 2	6
dayNight	display information about the period of a day given a time	32

includes a sub-part of **NewCaledonia-5690** composed of contributions associated with 8 exercises selected for their algorithmic diversity (see Table 2) and their balanced volumetry (100 to 150 programs per exercise). We will use it as a 'toy' dataset facilitating qualitative analyzes. The **Dublin-42487** dataset includes student programs from the University of Dublin, carried out between 2016 and 2019. Although the original corpus [3] contains nearly 600,000 programs (Python and Bash), we propose here a subset enriched semi-automatically with test cases (not provided initially).

4.2 Embedding analysis

In the following experiments, each dataset has been divided into three sub-parts: training (90%), validation (5%) and test (5%); the validation set being used to select the best model among those learned during the different iterations ($aes2vec$). Unless otherwise stated, the $aes2vec$ algorithm has been set up to learn embeddings of dimension 100, the size of the context is set to 2, concatenation is used as aggregation stage and the training is performed over 500 iterations.

In a first step, we evaluate our approach in a qualitative way on the dataset **NewCaledonia-1014** constituted for this purpose. Figure 4 (left) shows a visualization of the 912 programs of the training set, obtained by a non-linear projection using the t-SNE dimension reduction algorithm [9]. It can be seen that, although embeddings are learned in an unsupervised manner, the $code2aes2vec$ method learns, from a relatively limited number of training data, a representation space in which the areas identify distinct program functionalities. Thus, the program vectors are organized quite naturally into 8 clusters that are highly correlated with the original 8 exercises. Moreover, the topological organization of the clusters matches well with the algorithm inherent to the programs. Exercises 'swapping' and 'minimum' are close in the embedding space and correspond to the only two exercises that iterate over all values of an input list. Exercises 'compareStrings', 'fourMore100' and 'indexOccurrence', in

the upper part of the space, require to partially iterate over a list. Finally, the last three exercises do not require loops and rely only on the use of conditional instructions. The exercise 'dayNight' is distinguished by the expected use of a display function (`print`) while all the other exercises return a result (`return`). This feature may explain the 'isolation' of the programs from this exercise from other programs.

Stylistic differentiation of programs is difficult to assess quantitatively. This task would require either objective criteria that can be extracted automatically or expensive expert labeling. Given the absence of such stylistic knowledge in datasets, we choose to illustrate on an example how the *code2aes2vec* method distinguishes different styles in the writing of a same function. Figure 4 (right) presents in detail the embedding space learned for the exercise 'minimum'. It's interesting to notice that program styles are clearly distinguished, notably the two ways to program a Python `for` loop (using indexes vs. elements directly). In a very detailed way, programs are also grouped according to whether their loop starts with the first element of the list (`range(0,...)`) or with the second one (`range(1,...)`), after initialization of the minimum to the first element in any cases.

Word order is important in our *aes2vec* method. For example, our approach distinguishes programs having a similar boolean condition but expressed in a different order (`if liste[i]<res:` vs. `if res>liste[i]:`). This distinction may seem artificial since these two expressions are strictly equivalent from the evaluation point of view. However, the first syntax appears more 'natural' than the second. It would be easy to get rid of this phenomenon by normalizing the expressions at the *code2aes* step; this option can be left at teacher's discretion.

Finally, we draw the reader's attention on some valid but atypical programs. In particular a program using the native Python function `min`, or the one using the `sort` function. Their separation from the rest of the programs is crucial since it offers a way to detect programs (a priori valid) that a teacher would like to reject or at least moderate considering that they deviate from his/her pedagogical objective. More generally, this analysis seems to confirm that the embedding spaces learned by the *code2aes2vec* method correctly captures not only the function of the programs but also their style. Their intrinsic quality paves the way for many practical uses that could significantly improve the efficiency of learning platforms (detection of atypical solutions, automation/propagation of feedbacks, student 'trajectory' analysis, study of error typologies, etc.).

In a second step, we evaluate the *code2aes2vec* approach from a quantitative point of view on the three datasets (Table 3). We consider an usual task for program embedding evaluation, namely the prediction of its function (i.e. exercise identification). For each considered configuration (AES level), the training data are used first to learn (without supervision) a representation space. Then, these embeddings are used to learn (with supervision) a SVM classifier (with polynomial kernel) [5]. Finally, the embeddings are (indirectly) assessed according to their ability to predict the function of the code (i.e. the exercise) for test data.

As baselines, *random classifier* informs about the difficulty of this task *a priori*, while *doc2vec* corresponds to the (naive) use of the algorithm *doc2vec* [8] to learn embeddings from the codes directly (without any intermediate representation). We also report the results obtained by the (supervised) *code2vec* approach[5] [2] executed with default parameters.

	NC-1014	NC-5690	Dublin-42487
random classifier	*0.125*	*0.015*	*0.009*
code2vec [2]	0.230	0.098	0.037
doc2vec [8]+SVM	0.412	0.495	0.380
code2aes2vec+SVM			
(AES-0)	0.882	0.460	0.391
(AES-1)	**1.0**	0.698	0.544
(AES-2)	**1.0**	**0.832**	**0.651**

Table 3: Quantitative and comparative evaluation of the produced embeddings, on the task of retrieving the function of a program (accuracy).

It can be seen that the *code2vec* model recently proposed by [2] cannot be trained satisfactorily on any of the three datasets. This is due to the numerous parameters to learn and the large number of examples this method requires. In the largest dataset (Dublin-42487), *code2vec* "only" has 42,487 programs as inputs. To the opposite, our *code2aes2vec* method as several million entries thanks to our AES intermediate representation.

The comparative results obtained with three different levels of AES (denoted by AES-0, AES-1 and AES-2 in Table 3) confirm that the quality of the embeddings is improved when the level of detail of the AES increases. Level 2 AES (AES-2) are undeniably leading to the best vector representations of programs.

In order to take into account the word/token order, *code2aes2vec* and *doc2vec* have been set up so far with concatenation as aggregation step. In order to confirm the importance of the order, we compare in Figure 5 embeddings obtained by the *code2aes2vec* algorithm with both types of aggregation (sum vs. concatenation). Unlike for concatenation, we observe a very rapid degradation in the quality of the embeddings obtained with a sum type aggregation when the size of the context increases. Indeed, the vocabulary on which AESs are based is very limited (only a few dozen or even hundreds of tokens) and the distribution of these words in AESs is not uniform. Thus it quickly becomes difficult to differentiate contexts as their size increases without taking into account the word order.

4.3 Application to feedback propagation

In order to confirm that the learned embeddings are fine enough to be usefully exploited in an educational context, we have implemented a first proof of concept on the task of propagating feedbacks.

We have considered the exercise *'mean'* from the dataset **NewCaledonia-5690** whose instruction was to write a Python

[5]The other methods presented in the state of the art could not be compared because of the lack of available operational implementations.

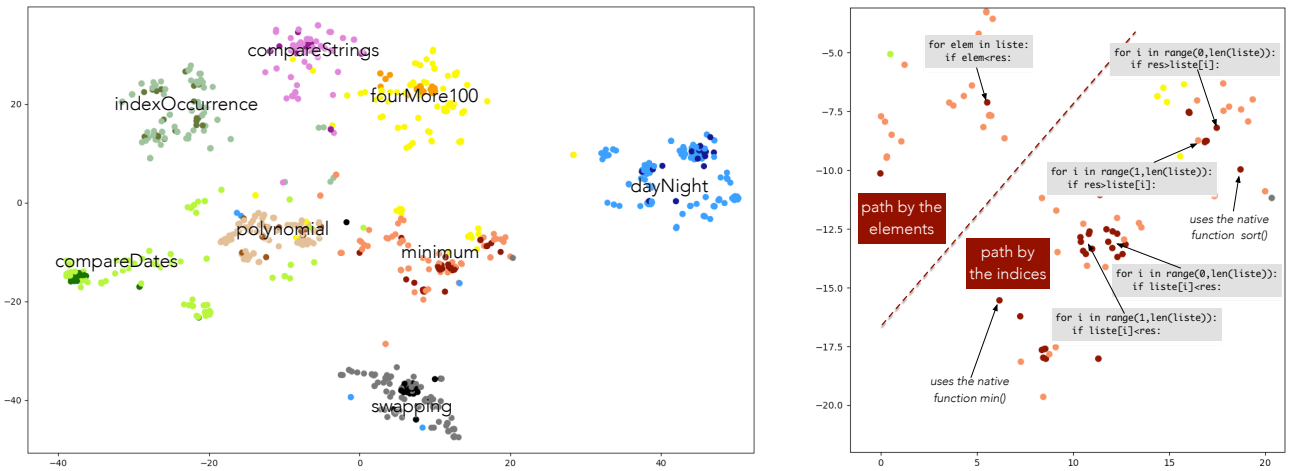


Figure 4: Visualization of program embeddings obtained by the method *code2aes2vec* for the 8 exercises of the dataset *NewCaledonia-1014*. The colors identify the exercises, with incorrect programs in light and correct ones in dark. The figure on the left represents all the embeddings and the one on the right details the area associated with the 'minimum' exercise.

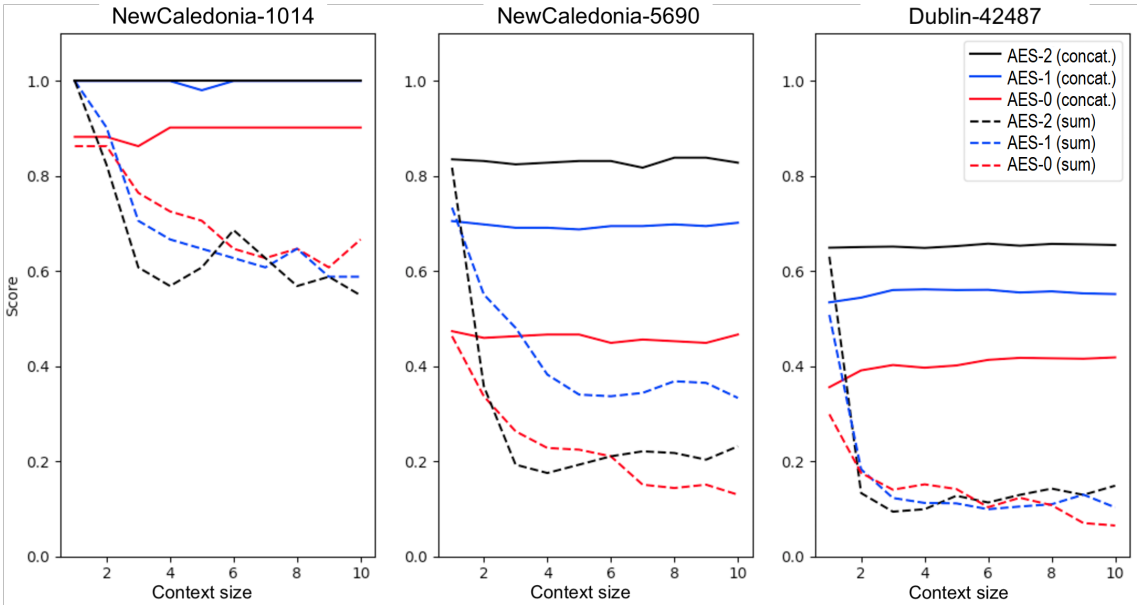


Figure 5: Evaluation of the embeddings on each of the three datasets according to the type of AES, the aggregation method and the context size (nb. words before/after).

function returning the mean of the values contained in a list passed as a parameter (and returning None if this list is empty). For this exercise, 157 programs were submitted to the platform by 24 different students ; among these submissions 122 programs were evaluated as incorrect by the platform and on which we sought to propagate teacher feedbacks based on their embeddings.

For this purpose, we performed a clustering (k-means [10]) on the 122 incorrect programs. We then presented to a teacher the most representative program (cluster medoid) of each cluster obtained, asking him to provide one feedback to help the student correct his proposal[6]. Once the k feedbacks were compiled (one per cluster/medoid), we went through each cluster and asked the teacher, for each program (other than the medoid), to indicate whether the feedback defined for the medoid could be applied to that other program. The objective is thus to assess the extent to which feedback from one medoid can propagate to all other programs in the same cluster.

Operationally, clustering is performed on the 122 incorrect

[6]If more than one error is found, the teacher must choose the one he/she feels needs to be corrected first. Each feedback is thus limited to the resolution of a single error.

programs, defined by their embeddings in $\mathbb{R}^{100}$. For a fixed number of k clusters, the partition selected to be analyzed is the one minimizing the MSE (Mean Square Error) among 100 runs (random initializations) of the k-means.

Table 4 presents the partition obtained for 5 clusters ($k = 5$). For each cluster we indicate its size, the program associated with its medoid as well as the feedback provided by the teacher for this medoid.

Let us first observe that the feedbacks provided by the teacher may relate to errors different in nature. It can be either an error in the design of the algorithm (clusters 1 and 3) or an error in the writing of the Python program (clusters 2, 4 and 5).

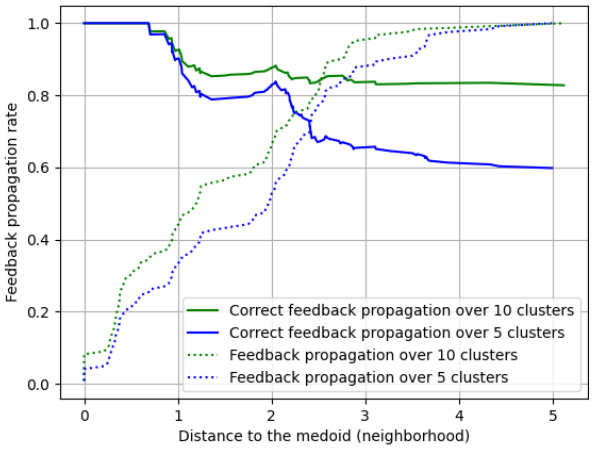


Figure 6: Propagation of teacher feedback to neighboring programs in each cluster. Illustration on the exercise *mean*.

We repeated this work with a partition into 10 clusters, this time asking the teacher to provide 10 feedbacks. Finally, we measured the rate of correct feedback when propagating the feedback from each medoid to its neighboring programs in the same cluster. Figure 6 shows the evolution of the correct feedback rate as a function of the neighborhood size considered for propagation. It can be seen that the further away from the medoids, the more errors in the automatically determined feedbacks.

Of course, a small number of feedbacks (5 or 10) is not enough to cover all the errors present in the 122 incorrect programs. In practice, it will therefore not be envisaged to propagate to all the programs in the cluster but only to the neighboring programs of the medoid. The dashed curves in Figure 6 indicate the proportion of programs covered as a function of the size of the neighborhood under consideration. We can see that with a neighborhood radius corresponding to a distance of 1.0, a propagation over 5 clusters allows to cover 33% of the programs with a precision of 90%; similarly such a propagation on 10 clusters allows to cover significantly more programs (43%) with a higher precision (93%).

The use of embeddings allows in this example to assist the teacher in his task of accompanying the students. For each feedback requested, 4 to 5 additional neighboring programs are automatically processed. Moreover, it is reasonable to think that over time a sufficiently large collection of feedbacks will be defined by the teacher to cover almost the entire embedding space so as to systematically identify one relevant feedback each time a new incorrect program is submitted and its embedding will position it in a pre-identified neighborhood.

5. CONCLUSION AND PERSPECTIVES

This paper studies the problem of learning vector representations, or embeddings, of programs in an educational context where the function is just as important as the style. Faced with this problem, we propose the method *code2aes2vec* transforming the code into abstract execution sequences (AES), and then into embeddings. This approach adapts the *doc2vec* method for program application and is based on the *document-based distributional variations* hypothesis.

The publication of the source code of the approach is accompanied by the availability to the community of a new enriched 'corpus' composed of more than 5,000 student programs (Python). Experiments conducted on these new data and on a public data set validate the quality of the learned embeddings, capturing in a fine way the function and the style of the programs. In addition a promising proof of concept was carried out on a classical task in the field, namely the propagation of teacher feedbacks.

The perspectives of this work are numerous. First, our experimentation focus on programs done in introductory courses, i.e. pretty simple codes. It would be interesting to analyze more elaborated ones (from more advanced courses) and to evaluate the impact of code complexity on performance.

Then, it seems necessary to complete the results observed on the stylistic differentiation of programs, by formalizing the notion of *style* of a program, in order to quantitatively evaluate our program embeddings. In the same way, a more precise analysis of the test cases used will have to be carried out in order to determine to what extent the constructed embeddings are sensitive to them.

Finally, to have more exploitable corpora, we plan to extend our implementation to handle any type of language (the current implementation only processes the Python language).

From a more methodological perspective, all the words in the program have the same weight during the embedding construction in our approach. Thus, a correct program and one returning a wrong value (or throwing an error) may have very similar embeddings, although functionally very different. This aspect could be integrated in the construction of our AES or in the architecture of the neural network used to generate embeddings. For that, it could also be interesting to add to our AES the values taken by the variables, in the same way that [17] but in a generic multimodal approach. Another perspective would be to allow the expert to integrate part of his knowledge on the language. As discussed previously, some instruction sequences can be equivalent

Cluster 1 (#31)	**Cluster 2** (#22)	**Cluster 3** (#28)	**Cluster 4** (#9)	**Cluster 5** (#32)

`def mean(l):`
`  if len(l)==0:`
`  res=None`
`  else:`
`  res=0`
`  cpt=0`
`  for elem in l:`
`  res=res+elem`
`  cpt=cpt+1`
`  res=elem/cpt`
`  return res`
	`def mean(l):`
	`  if len(l)==0:`
	`  res=None`
	`  else:`
	`  s=0`
	`  for elem in l:`
	`  s+=elem`
	`  res=s//len(l)`
	`  return res`
		`def mean(l):`
		`  if len(l)==0:`
		`  res=None`
		`  else:`
		`  res=0`
		`  cpt=0`
		`  avg=0`
		`  for elem in l:`
		`  res=res+elem`
		`  cpt=cpt+1`
		`  avg=res/cpt`
		`  return avg`
			`def mean(l):`
			`  if l==():`
			`  res=none`
			`  else:`
			`  res=0`
			`  for i in range(len(l)):`
			`  x=res+l[i]`
			`  res=x/len(l)`
			`  return res`
				`def mean(l):`
				`  if len(l)==0:`
				`  res=None`
				`  else:`
				`  res=0`
				`  cpt=0`
				`  for elem in l:`
				`  res=res+elem`
				`  cpt=cpt+2`
				`  res=res%cpt`
				`  return res`

The division step must be performed once the sum calculation is completed (put this instruction out of the `for` loop).	*The `//` operator corresponds in Python to the integer division. For the computation of a mean a simple division is required (operator `/`).*	*In the case of an empty list, your function does not return None (as requested).*	*The null value in Python is written 'None' (instead of 'none').*	*The `%` operator corresponds in Python to the modulus. For the computation of a mean a simple division is required (operator `/`).*

Table 4: Description of the 5 cluster partition generated by k-means on the embeddings of the incorrect programs from the *mean* exercise. For each cluster (table column): (1) the number of programs, (2) the program associated with the medoid of the cluster and (3) the feedback defined by the teacher for this program. Instructions in red are the ones that are questioned in the feedback.

(e.g., `if liste[i] <res:` vs. `if res> liste[i]:`). Semantic relations between words can also be known (e.g., the relation between `for` and `while` statements). This knowledge could be used to constrain the neural network and guide embedding construction. Finally, these program embeddings open up a large number of perspectives for teaching aid, in addition to the task of feedback propagation. For example, they could be used to identify error typologies, alternative solutions, or even predict dropout students through the analysis of their 'trajectories'.

6. REFERENCES

[1] M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton. A survey of machine learning for big code and naturalness. *ACM Computing Surveys*, 51(4):1–37, 2018.

[2] U. Alon, M. Zilberstein, O. Levy, and E. Yahav. code2vec: Learning distributed representations of code. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29, 2019.

[3] D. Azcona, P. Arora, I.-H. Hsiao, and A. Smeaton. user2code2vec: Embeddings for profiling students based on distributional representations of source code. In *Proceedings of the International Conference on Learning Analytics & Knowledge*, pages 86–95, 2019.

[4] R. Bazzocchi, M. Flemming, and L. Zhang. Analyzing cs1 student code using code embeddings. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 1293–1293, 2020.

[5] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[6] D. DeFreez, A. V. Thakur, and C. Rubio-González. Path-based function embedding and its application to error-handling specification mining. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 423–433, 2018.

[7] J. Henkel, S. K. Lahiri, B. Liblit, and T. Reps. Code vectors: understanding programs through embedded abstracted symbolic traces. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 163–174, 2018.

[8] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[9] M. LJPvd and G. Hinton. Visualizing high-dimensional data using t-sne. *J Mach Learn Res*, 9:2579–2605, 2008.

[10] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[12] T. D. Nguyen, A. T. Nguyen, H. D. Phan, and T. N. Nguyen. Exploring api embedding for api usages and applications. In *IEEE/ACM International Conference on Software Engineering*, pages 438–449. IEEE, 2017.

[13] H. Peng, L. Mou, G. Li, Y. Liu, L. Zhang, and Z. Jin. Building program vector representations for deep learning. In *International Conference on Knowledge Science, Engineering and Management*, pages 547–553. Springer, 2015.

[14] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[15] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas. Learning program embeddings to propagate feedback on student code. In *Proceedings of the 32nd International Conference on Machine Learning*, ICML'15, page 1093–1102, 2015.

[16] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[17] K. Wang, R. Singh, and Z. Su. Dynamic neural program embeddings for program repair. In *International Conference on Learning Representations*, 2018.

Student-centric Model of Login Patterns: A Case Study with Learning Management Systems

Varun Mandalapu[1], Lujie Karen Chen[1], Zhiyuan Chen[1], Jiaqi Gong[2]
[1]University of Maryland Baltimore County, Baltimore, Maryland, USA, 21250
[2]The University of Alabama, Tuscaloosa, Alabama, USA, 35487
{varunm1, lujiec, zhchen}@umbc.edu, jiaqi.gong@ua.edu

ABSTRACT

With the increasing adoption of Learning Management Systems (LMS) in colleges and universities, research in exploring the interaction data captured by these systems is promising in developing a better learning environment and improving teaching practice. Most of these research efforts focused on course-level variables to predict student performance in specific courses. However, these research findings for individual courses are limited to develop beneficial pedagogical interventions at the student level because students often have multiple courses simultaneously. This paper argues that student-centric models will provide systematic insights into students' learning behavior to develop effective teaching practice. This study analyzed 1651 undergraduate student's data collected in Fall 2019 from computer science and information systems departments at a US university that actively uses Blackboard as an LMS. The experimental results demonstrated the prediction performance of student-centric models and explained the influence of various predictors related to login volumes, login regularity, login chronotypes, and demographics on predictive models. Our findings show that student prior performance and normalized student login volume across courses significantly impact student performance models. We also observe that regularity in student logins has a significant influence on low performing students and students from minority races. Based on these findings, the implications were discussed to develop potential teaching practices for these students.

Keywords

Student-centric Modeling, Learning Management Systems, Login Variables, Student Performance Prediction.

1. INTRODUCTION

Teaching and learning changed a lot in recent years with the increasing adoption of new computer-based teaching and learning technologies in educational institutions worldwide. As education and learning technology evolves with time, leveraging the technical advances to improve teaching practice and student learning will be a prominent research area. The most common technologies used by instructors to deliver course content include Learning Management System (LMS), Course Management Systems (CMS), and Learning Content Management Systems (LCMS) [1]. Even though these systems seem to be synonymous, they have their specific use in the education domain. LMS tools focus on communication, collaboration, content delivery, and assessment, whereas LCMS is similar to LMS with fewer administrative functions. CMS, on the other hand, will focus on the enrollment and performance of students. Of these three systems, LMS is the one that is best suitable for delivering learning strategy to students and is the primary focus of this study.

LMS systems provide a unique opportunity to administrators, and researchers to evaluate student data related to time spent on an activity, access times and day, grades, interactions, and many other useful student learning variables. The data logs collected by LMS systems are analyzed with scientific techniques published in the Educational Data Mining (EDM) domain. In their study, Romero and Ventura [2] described that current EDM methods rely on clustering and pattern recognition techniques to categorize students into various groups based on their interaction patterns. Categorization of students using clustering and pattern recognition supports instructors in making changes for a set of students. Teaching practices that impact the entire classroom can be evaluated using predictive analytics that tracks student learning and achievement from the vast amount of interaction data collected by LMS.

Existing research in Learning Analytics (LA) and EDM focused on developing highly accurate predictive models that can estimate student learning outcomes related to assignment scores, course grades, and drop-out probability [3,4]. These course-based predictive models provide early warning to student counselors or instructors associated with a specific course [5,6]. Even with considerable success in this area, many of the student performance prediction models have several shortcomings. One significant issue with course-based models is the bias introduced by teaching style and the type of course (descriptive, programming, mathematical etc.). This bias impacts these models' scalability across different courses and makes it difficult to understand the student level factors on their achievement. For example, if a student enrolls in five courses, developing models to study students' progress in these five courses independently is not realistic and gives different insights based on varying features and performances. Therefore, these modeling efforts are limited to reduce different biases introduced by instructor and the diverse amount of content made available in LMS.

Course level predictions are suitable for supporting instructor level decision making; however, if intervention is on student level behaviors such as study habits or self-regulation skills, it is beneficial to look at student-centered indicators so that interventions may be more targeted and cost-effective [7,8]. Developing student-centric models that analyze student LMS interactions across courses in a college/university setting will help

Varun Mandalapu, Lujie Chen, Zhiyuan Chen and Jiaqi Gong "Student-centric Model of Login Patterns: A Case Study with Learning Management Systems". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 263-274. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

address the issues with course specific models. This study is the first step in developing models that supports the identification of student level indicators.

For colleges that have a high penetration of LMS, LMS activity may give a holistic indicator of students' engagement level (behavior engagement specifically). We ask the question to what extent those holistic indicators predict student term Grade Point Average (GPA) performance in the future. To explore this, we specifically focus on student login related features as they can be generalized across courses and act as proxy variables for time management [51, 52]. It is also challenging to aggregate other features like discussions, readings, and assessments across courses compared to access related LMS variables. This study's data is drawn from Blackboard Learn, a commercial LMS software available for colleges and universities to deliver course content and assessments through internet-enabled computer systems. Most importantly, the data is drawn from all students in computer science and information systems at a large public university in the US during the Fall 2019 semester. In addition to student interactions from LMS, we also access demographic and prior student performance data from the university's student administration system to build and interpret downstream predictive models. The university's Institutional Review Board (IRB) approved this study, and all the student specific demographic and personal information are anonymized by following General Data Protection Regulation (GDPR) standards.

In this work, we focus on model predictions and explanations to understand student learning behaviors. First, we apply new methods to process student interaction data collected across different courses enrolled in a semester to build student-centric performance models based on machine learning principles. Secondly, we utilize a novel approach in local model explanations, correlation and regression to understand the impact of various features captured by LMS on student performance. One primary reason for using Locally Interpretable Model Explanations (LIME) is its ability to explain the relationship between predictor variables and predictions, especially the input variable's impact on the outcome. On the other hand, statistical correlation analysis will provide the relation between input predictors and the observed target variable. As correlation analysis does not consider the interaction effect between input variables, we also use a linear regression model to study the output variable's feature importance's based on the model coefficients. To address the research gap discussed earlier, we explore the below three research questions.

RQ 1 How different student-centric machine learning models perform in predicting student end-of-term GPA?

RQ 2 How do student login and time interval pattern across courses influence student learning outcomes?

RQ 3 Is there a significant variability in feature importance for students coming from diverse demographics?

2. RELATED WORK

Universities and colleges around the world adopted LMS systems, such as Moodle and Blackboard, to provide onsite, hybrid, and online courses based on their capabilities to support communication, content creation, administration, and assessment [9, 10]. Besides the automation and centralization of various administrative tasks like creating and managing student accounts, creating syllabus, assignments, assessments, grading, etc., LMS systems assemble and deliver personalized learning materials and content quickly [11]. These systems also support the reusability of materials created by instructors. The systems also enable instructors to create content structures, deliver them in a sequence, maintain control access, organize group activities, track student activities, load and replace learning materials and provide feedback on assessments. With advanced database software developed by Oracle, IBM, and Microsoft that emphasize interconnectedness, data independence, and security, LMS systems employ various login roles based on user classification. These roles will permit instructors to create new content or privately address student issues and create discussion boards to capture student knowledge on specific topics.

LMS platforms enable students to access learning material in various formats, such as pdf, PowerPoint presentations, video lectures, and audio files. The systems also track student activity related to content downloads, access timestamps to display student progress in learning to instructors [12]. LMS also provides both asynchronous and synchronous communication for students to interact with instructors and encourages group activities. Combining the tools provided by LMS with innovative learning strategies like self-directed learning, small group instructions, and collaborative learning with instructor interventions, a wide variety of activities can be developed for individual, small groups, or larger classes [12,13]. Given the simplicity and convenience of accessing online materials through LMS systems, it is not surprising to see high student satisfaction scores for courses delivered through LMS.

In recent years, the data sets related to student learning activities have drawn significant attention from researchers in academic communities to develop possible solutions to address student retention and academic success issues. This type of work has been called learning analytics and focuses on student activities such as navigating lecture materials, what information is accessed, how long it takes to complete an activity, and how students transform the information in learning materials into measurable learning [14, 15]. Multiple commercial resources like SPSS, google analytics, Stata and Nvivo can build predictive models on data captured by LMS to assess student drop-out probabilities to develop targeted learning courses or model collective learning behaviors. Since most instructors deliver course assessments and material through LMS, they can track student activity by processing a digital footprint during every online interaction captured by system log files.

2.1 Learning Analytics Research

LMS systems have the ability to capture large data streams related to user interactions through which administrators and instructors can develop methods to improve the learning experience. The collection, analysis, and reporting of data about learning activities on web-enabled learning platforms to assess student academic progress, predict performance, and identify potential issues that need attention is the central proposition of emerging fields like learning analytics and educational data mining [16, 17]. Outcomes derived from learning analytics aim to gain insights about student learning behaviors, real-time information about institutional practices and support the designing of personalized courses in CMS. Although there are huge data stores in universities and colleges that can be used to make data-driven decisions to support optimal use of both pedagogical and economic resources, to date there has been minimal application of this data in higher education [18].

2.1.1 Student Engagement and Frequency of LMS Use

An LMS system records student interaction details related to logins, number of posts written on discussion threads, time spent on lecture materials, total downloads, etc., in their log files. These logs can be analyzed to generate reports that help teachers to observe student progress at a granular level. Once there are enough student records collected in LMS, they can be used to develop computational models to predict future student performances. Multiple works in EDM and LA studied the relation between the usage of LMS and student academic achievements. Vengroff and Bourbeau's [19] study showed evidence that providing additional material in LMS benefited students at the undergraduate level. They also conclude that students who used LMS regularly did better in exams than their peers who have minimal interactions. In their research, Dutt and Ismail [20] observed that tracking resources students interact with on LMS supports developing new strategies that make learning easier and enhance learner progress. Their work also focused on analyzing thresholds related to student interaction features like self-assessment tests, time spent on exercises, discussion forums, and performance outcomes. Another study by Lust et al. [21] explored the usage variations in different tools used by students on LMS, such as time on web-link, time on web-lectures, time on a quiz, time on feedback, postings on discussion board, and messages read. The results from this study heavily contributed to the development of adaptive and innovative recommendation systems. In their work, Hung and Zhang [22] also found patterns based on six indices that represent student effort: Frequency of accessing the course material, number of LMS logins, total interactions in discussion threads, number of synchronous discussions, number of posts read, and final grades in a course.

While exploring a link between student online activity on LMS and their grades, Dawson et al. [23] observed a significant difference in the number of online sessions accessed, total time spent, and the number of posts in discussion forums between high and low performing students. Another study by Damainov et al. [24] developed a multinomial logistic regression model based on time spent in LMS. This study found a significant relationship between student time spent and grades, especially in students who attained lower grades between D and B. Instead of using time spent online, other works focused on the frequency of course material access within LMS. A study by Baugher et al. [25] found that regularity in student hits is a reliable predictor of student performance compared to the total number of hits. In their study, Chancery and Haque analyzed student interaction logs of 112 undergraduate students and found students with low LMS access rates obtained lower grades than their peers with higher access rates. This study was complemented by Biktimirovan and Klassen [26] that reported a strong relationship between student hit consistency and success. Their study counted access to various LMS activities and found that homework solution access is the only strong predictor of student performance. However, these studies are primarily descriptive rather than predictive.

2.1.2 Instructional Design and Student Participation

Online teaching strategies are primarily dependent on instruction design as each mode of interaction - student/instructor, student/student, and student/content have their own positive impacts on student progress. A study by Coldwell et al. [27] focused on the relationship between student participation in a fully online course and their final grades. They found a positive relationship between student participation and final grade. Dawson et al. [23] examined the impact of various LMS tools and found a highly positive correlation between discussion forum activity and student success. They observed more than 80% of interactions occurred in the discussion forum, which is the primary interaction tool in LMS. Another study by Greenland [28] found that asynchronous communication is the primary form of all online course interactions. Nandi et al. [29] found an increasing number of posts in discussion forums close to assignment and exam deadlines. They also found a high correlation between exam scores and online class participation throughout the semester, especially in high-achieving students.

All the studies discussed above adopted log files from LMS systems to extract unbiased details from activity and performance to identify a relationship between independent interaction variables and student grades. Most of the discussed studies are based on univariate analysis focusing on a single variable or a set of highly impactful variables of a single course or similar courses on student outcomes. However, student performance is a highly complex area in education to measure or understand, especially across various courses offered on-campus in a university setting. Most of the authors discussed above noted the need for more in-depth works to investigate student performance across courses and based on multiple variables. These studies also lack an explanation about variables used in their studies to track student performance, and it is evident that the authors selected LMS variables based on their belief that these variables are highly correlated with student scores.

2.1.3 Social Factors in Analytics

Factors that influence student academic performance have been the focus of researchers in LA and EDM domains for many years. It still remains an active area of education research, indicating the complex problem in measuring and modeling learner processes, especially in tertiary education. Positive learning characteristics have a significant positive impact on learner engagement improvement in multiple ways. The dispositional language specifies learning as a combination of self-regulation, learning inclinations, motivation, behavioral patterns, interactions, and cognitive ability. In their study, Buckingham et al. [30] proposed a combination of self-reported data gathered in surveys with student interaction data generated by LMS to study individual student performance, learning processes, and group interactions. These social analytics depend primarily on student self-reported data to develop toolkits that support a specific learning type, especially in courses with high diversity [31]. However, our study focuses on objective identification of student success based on data that LMS captures. We will also identify the crucial variables from predictive model output for various student groups based on their diverse backgrounds (race, gender, and student status).

2.1.4 Multivariate Analysis to Predict Student Success

Even though there is a common agreement about the purpose of learning analytics, there are still several varying opinions on what data needs to be collected and analyzed to improve teaching and learning processes. A study by Agudo-Peregrina et al. [32] argued that it is highly complex to identify the net contribution of various interactions to the learning processes. Their findings show that peer interaction between students has a lower influence than student-teacher interaction, which contradicts earlier studies that showed high importance for student peer interactions. A study by Dominquez et al. [33] utilized multiple variables like LMS logins, time stamps, and content access flags captured in a biology course

to predict student grade at the end of course completion. The results show that the algorithm predictive accuracy is at 50% in subsequent semesters. Lerche and Keil's [34] recent study utilized Moodle log data from 369 students enrolled in three online courses across three semesters to predict their scores at the end of the term for each course. Their regression results related to predicting student scores in a course at the end of the semester varied from 0.17 to 0.6 for all three courses. This broad range of performance across courses is due to varying variables utilized in each course based on the course structures. Studying the difference in instructional design, variables in extracted data, statistical inferences, predictive modeling used, interpreting model outcomes and pattern observations, etc., might explain the inconsistencies in results shown in earlier studies.

Data captured by LMS systems became prominent in LA and EDM circles as they capture student interactions in non-intrusive and ready-to-use settings. Several studies were discussed earlier in this research that utilized the LMS data to develop models that track student progress. However, it is still challenging to build highly accurate models that predict student learning outcomes across courses and understand the impact of different variables captured by LMS. Another significant gap in earlier research is their inability to predict student performance across courses in a given semester. One primary issue in predicting student performance across a semester is to find methods that aggregate student LMS variables across courses. This research shows methods to address the research gap found in earlier studies.

In this study, we approach the problem of tracking student achievement by developing student-centric models that build on aggregated LMS interaction variables collected across a semester irrespective of student year and course. One unique aspect of our work is related to the study of model performance on longitudinal student data. We develop models that predict student end-of-term GPA based on four cumulative periods in a semester. This work also focuses on explaining the impact of different aggregated LMS variables on various student groups categorized based on performance, race, gender, and student type. The importance of features is explained by adopting correlation statistics for univariate importance, a regression model for interaction effect, and LIME for model-based yet model agnostic explanations.

3. DATA & FEATURE SET

3.1 Dataset

For this study, we chose undergraduate student data captured by LMS in Fall 2019 from a large public university in the United States. These students were part of either Information Systems (IS) or Computer Science (CS) departments. The students from these departments were chosen as the instruction format and courses are closely aligned in both of them. Blackboard system is predominantly used as an LMS to deliver course material, assessment, and grading. The student demographic data captured by a standalone Student Information System (SIS) is used to categorize students based on different demographic variables. A total of 1651 students were enrolled in these two departments in the Fall 2019 semester. Based on student distribution, we categorized students into three ethnicities: White, Asian, and Minority. This study also researches student performance based on their admit types, such as four-year regular student or transfer student. The demographics of student data are provided in the below table 1. This study was approved by IRB and sensitive student data was de-identified based on GDPR standards.

Table 1. Student demographics

Demographic	Student Count
Total Students (N)	1651
No of unique courses	440
No of unique course instructor combinations	638
Male : Female	1302 (79%) : 369 (21%)
White : Asian : Minority	630 (38%) : 495 (30%) : 526 (32%)
4 – Year : Transfer	976 (59%) : 675 (41%)
Full Time : Part Time	1446 (88%) : 205 (12%)
IS : CS	934 (57%) : 717 (43%)
1st Yr : 2nd Yr : 3rd Yr : 4th Yr	115 (7%) : 329 (20%) : 515 (31%) : 692 (42%)
<= 3 : 4-5 : >5 (Courses enrolled)	298 (18%) : 1035 (63%) : 318 (19%)

3.2 Feature Extraction

We explored various LMS features related to student logins, content accesses, time spent, discussion posts, assignment submissions, and time intervals based on earlier literature. While exploring these features, we identified that only three features could be commonly extracted from different courses: Student Login Counts, Time intervals & prior knowledge.

One of the significant challenges while building a student-centric model on LMS data is to extract aggregated features that are least biased. As Blackboard's content is dependent on instructor and course, it is crucial to mitigate the variations caused by these factors on aggregate student variables. This work employs multiple statistical measures to mitigate these issues. The details are explained in the below sub-sections.

3.2.1 Normalized Login Volume

Earlier studies identified that student performance prediction is strongly dependent on the volume of student logins. One challenge with counting the student logins in Blackboard is its inability to find which course they accessed during each login. Also, calculating the total login count introduces a hidden bias as courses with more content on Blackboard prompt students to login more often than other courses with less content and flexible deadlines. To mitigate this issue, our work followed the below steps to extract student login features.

1. Extract all courses enrolled by all students in IS and CS.
2. Count the total number of logins for all students irrespective of their department in these extracted courses.
3. Calculate the Z-scores of student logins in each course. The reason for doing this is to mitigate the bias introduced by variations in the absolute count of logins as course logins vary a lot between students. Z-scores provide a value that helps understand if student logins are higher or less than average logins in a specific course.

4. Once the z-scores are calculated for all courses, we extract a vector of login z-scores for each student based on their enrolled courses.

5. As predictive models do not take vectors of variable length as input, this work extracts seven significant statistics from the login vector: mean, median, minimum, maximum, standard deviation, skewness, and kurtosis.

3.2.2 Login Regularity

Apart from student login volumes, the regularity between logins also provides valuable insights into student achievement as regularity is related to self-regulation capabilities. In this work, we utilize an entropy-based method to extract features that define student login regularity in each course. In information theory, entropy is used to define uncertainty or randomness [48]. Entropy measure will explain if student's logins are regular (less random) or irregular (more random). Based on this concept, if the entropy value is high, then a student has an irregular login pattern, and if the entropy value is low, the student has a regular login pattern. The steps to calculate student regularity features are given below.

1. Extract all course accesses with timestamps for every student in IS and CS.

2. Calculate the difference between timestamps. This difference will give a vector of time intervals for each course enrolled by a student.

3. Calculate entropy using the KL estimator with the k-nearest neighbor method proposed by Kozachenko and Leonenko [45]. KL estimator uses k-nearest neighbor distances to compute the entropy of distributions. The reason for adopting this method instead of Shannon entropy is based on the time interval vector's continuous characteristic [46].

4. Once the entropies are calculated, we get a vector of entropies for each student based on the number of enrolled courses. We then calculate the seven statistics similar to student logins: mean, median, minimum, maximum, standard deviation, skewness, and kurtosis.

3.2.3 Login Chronotypes

Studies in chronobiology and chronopsychology showed variation in different individual active periods at different times of the day [41, 42]. These studies classify an individual into either morning type or evening type based on their high activity time. For example, if an individual is highly active in the morning compared to the evening, they are considered morning type and vice versa. Inspired by this work in human psychology, this work divides a day into four-time bands T1 (12 AM to 6 AM), T2 (6 AM to 12 PM), T3 (12 PM to 6 PM), and T4 (6 PM to 12 AM) and extract student logins based on these four time bands. In addition to this, this work also extracts the logins on weekdays and weekends to study their influence on student performance.

1. Count the number of logins during each time band and on weekdays and weekends for each course.

2. Calculate the mean of login count vector for each of these time bands and weekday/weekend.

3. Normalize the login count with the number of courses enrolled by an individual student. This normalization will mitigate the bias introduced by the number of courses enrolled across the student cohort.

This work also utilizes the demographic and prior performance measured by GPA features captured by the SIS system. These features were listed in below table 2.

Table 2. Student demographic features

Demographic	Values
Start GPA (Prior Performance)	Cumulative GPA available till the start of semester
Gender	Male & Female
Ethnicity	White, Asian & Minority
Student Year	Freshman, Sophomore, Junior & Senior
Admit Type	Regular & Transfer
Enrollment Type	Full time & Part time
Student Age	Continuous variable

4. METHODOLOGY

The methodology section details the predictive modeling approach to predict student end-of-term GPA in fall 2019. In addition to this, we also describe the correlation-based LIME method to explain the features that contribute to model predictions. The workflow of developing student-centric models is depicted in figure 1.

4.1 Predictive Modeling

This work studied five of the most common regression models for comparison purposes. The selected models include Generalized Linear Model (GLM), Decision Tree (DT), Support Vector Regressor (SVR), Random Forest (RF), and Gradient Boosted Regressor (GBR). As model hyperparameter influences their predictive performance, we utilized a grid search mechanism to select multiple parameters to predict with high accuracy. We also adopted a feature selection method based on a multi-objective evolutionary algorithm in addition to hyperparameter search. This feature selection algorithm evaluates each feature set based on pareto-optimal that balances model complexity and accuracy. The details of models and hyperparameter search criteria are discussed below.

Generalized Linear Model: GLM is an extension of traditional linear models that fits input data by maximizing the log-likelihood. The regularization parameter is set so that the hyperparameter search space looks for an alpha value that fits between ridge and lasso regression. An alpha value of 1 represents lasso regression, and an alpha value of 0 represents ridge regression. This study searched for the best alpha value using a grid search between 0 and 1 in increments of 0.1.

Decision Tree: The decision tree algorithm is a collection of linked nodes intended to estimate the numerical target variable. Each node in the tree represents a rule used to split on an attribute value. The node uses a least-squares criterion to minimize the squared distance between the average value in a node when compared to the actual value. The hyperparameter search space for this algorithm evaluates both maximal depth and pruning. The maximal depth value varies between 1 and 100 in increments of 10. Pruning will make the DT algorithm use multiple criteria like

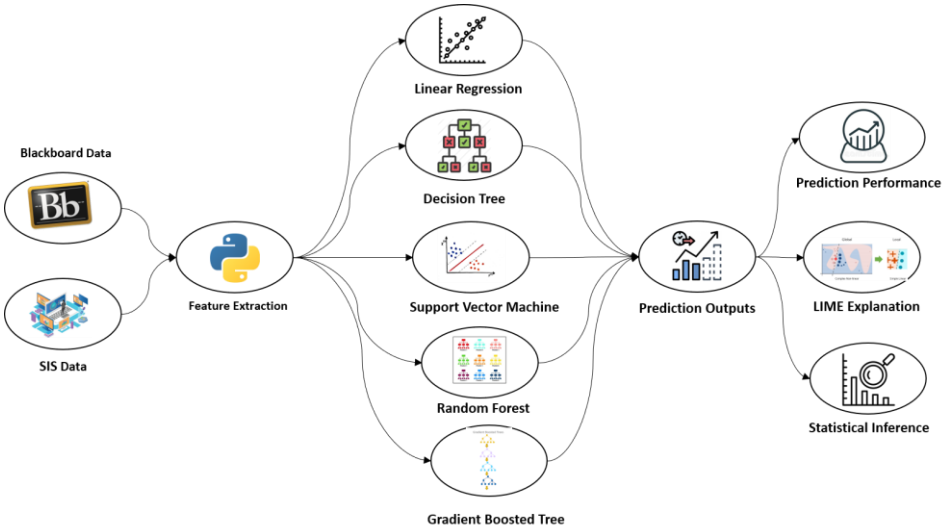


Figure 1: Student-centric Model Workflow

minimal gain, minimal leaf size, and pruning alternatives to decide the stopping criterion.

Support Vector Machines: The SVM used in this study is built based on Stefan Reupping's mySVM [47]. This algorithm will construct a set of hyperplanes in a high dimensional space for regression tasks. A good hyperplane is decided based on the functional margin. The hyperparameter search space focused on both dot and radial kernel functions with a C (SVM complexity) value range between 10 and 200. The kernel gamma function is set for a radial kernel with a range of 0.005 and 5 with three logarithmic increments.

Random Forest: A RF model builds an ensemble of decision trees on bootstrapped datasets. The splitting criteria are similar to a decision tree. The regression outcome is the average of the observed train data GPA present at that end node. We only tuned the number of trees hyperparameter to reduce the time complexity of the execution. The number of tree searches varied between 10 and 1000 trees in 10 linear steps.

Gradient Boosted Tree: The GBT model builds multiple regression trees in a sequence by employing boosting method. By sequentially applying weak learners on incrementally changed data, the algorithm builds a series of decision trees that produce and an ensemble of weak regression models. As GBT is a non-linear model, we search hyperparameters related to the number of trees, learning rate, and maximal depth. The number of tree values varies between 1 and 1000 in five quadratic increments, the learning rate varies between 0.001 and 0.01 in five logarithmic increments, and the maximal depth parameter varies between 3 and 15 in three logarithmic increments.

4.2 LIME Explanation

The concept of Locally Interpretable Model Explanations (LIME) was introduced to explain the predictions made by black-box models that deal with classification problems. LIME explains each prediction made by a complex model by training a surrogate model locally [35]. However, this earlier methodology is not scalable to deal with categorical variables, tabular data, and regression problems. In this work, we adopt the correlation-based LIME method available in RapidMiner to explain machine learning models' predictions [36, 37, 38].

1. Perturb data in the neighborhood of each sample in the dataset. The number of simulated samples can be user-defined. A higher number of simulated samples will provide higher accuracy of explanations but at the cost of more run times.
2. Make predictions using the ML model for all the simulated samples around each original sample in the dataset.
3. Calculate the correlation between each feature in the dataset and the target variable.
4. The features that have a positive correlation are considered supporting features, and features with negative correlation with predicted outputs are referred to as contradicting features.

As LIME provides feature importance value for each feature at each sample, we aggregate the importance value for all samples to build global importance for each variable. The significant advantage of this method compared to traditional global importance methods is its flexibility. As model global importance's are calculated across all samples in the data, the LIME based feature importance's can be calculated for subsets of data. This flexibility provides users with a deeper understanding of each feature's role for different sets of populations present in a dataset.

In addition to applying the LIME methodology, this work also studies univariate and multivariate feature importance on student performances by applying correlation and linear regression methods. The student dataset used in this study is divided into multiple subsets containing different student groups based on various demographics. A correlation value is calculated between input features and student end-of-term GPA. This value provides us with an intuition about the impact of various features on student performances related to different demographics. As correlation only provides independent variable importance on student performance, we also adopt a linear regression model to explore the variation of feature importance based on coefficient values. Applying a linear regression model will also consider the interaction effect between input features to fit the outcome variable.

5. RESULTS

This results section is divided into three subsections based on the three research questions we are focusing on in this study. The first subsection will detail various predictive models' performance on longitudinal student interaction data collected during the fall 2019 semester. The second subsection will detail the importance of student logins and regularity on performance predictions based on LIME methodology. The final subsection will discuss the importance of input features based on correlation and regression methods.

5.1 How different student-centric machine learning models perform in predicting student end-of-term GPA?

The five machine learning models adopted in this study were evaluated using a five-fold cross-validation method. In this method, the student data is divided into five equal folds at a student level. In every iteration, four of the five folds are used for model training, and one fold is used for model testing. The machine learning models are evaluated based on two performance metrics: R squared (R^2) and Root Mean Squared Error (RMSE). The output performance metrics are the average of five test fold performances.

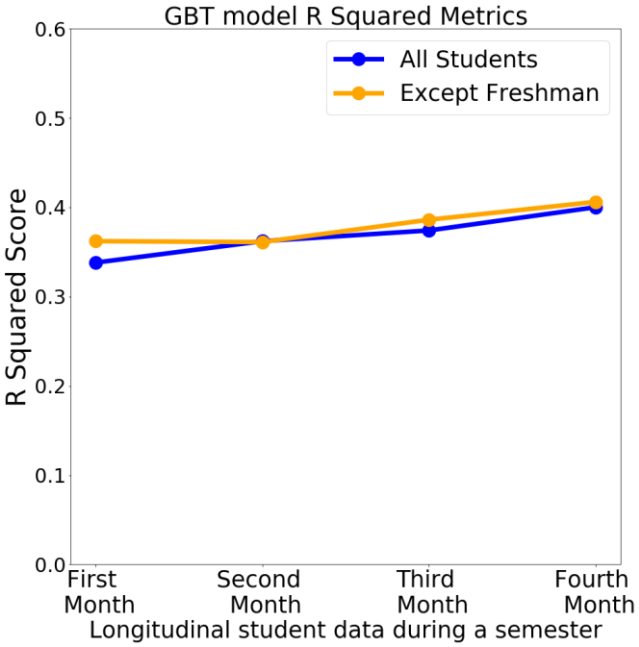


Figure 2. Compare performances of GBT model on different longitudinal datasets

In this study, we divided a semester into four parts to understand the impact of longitudinal interaction data across the semester on predictive model performances. This analysis will support the amount of data needed to balance predictive performance and early detection for interventions. The performance metrics evaluated on these four cumulative datasets will help understand the amount of student data needed to make accurate predictions. Tables 3, 4, 5, and 6 present the machine learning models' results evaluated on four cumulative datasets. While differentiating student performance based on multiple longitudinal datasets, we also study algorithms' performance without Freshman student data. This differentiation is to study the impact of missing start GPA feature values for first-year students as most of the full-time regular students in US universities start in the Fall semester.

Table 3. Student features from start to end of first month

Model	R^2		RMSE	
	All Students	Except Freshman	All Students	Except Freshman
GLM	0.213	0.249	0.657	0.633
DT	0.266	0.270	0.638	0.633
SVM	0.216	0.324	0.666	0.607
RF	0.332	0.353	0.607	0.588
GBT	**0.338**	**0.362**	**0.602**	**0.581**

Table 4. Student features from start to middle of semester

Model	R^2		RMSE	
	All Students	Except Freshman	All Students	Except Freshman
GLM	0.257	0.266	0.67	0.628
DT	0.263	0.295	0.67	0.618
SVM	0.195	0.315	0.705	0.609
RF	0.360	0.352	0.621	0.591
GBT	**0.362**	**0.361**	**0.622**	**0.586**

Table 5. Student features from start to end of third month

Model	R^2		RMSE	
	All Students	Except Freshman	All Students	Except Freshman
GLM	0.25	0.266	0.644	0.626
DT	0.255	0.255	0.658	0.650
SVM	0.335	0.344	0.612	0.597
RF	0.371	0.386	0.589	0.575
GBT	**0.374**	**0.386**	**0.588**	**0.572**

Table 6. Student features from start to end of semester

Model	R^2		RMSE	
	All Students	Except Freshman	All Students	Except Freshman
GLM	0.251	0.269	0.644	0.625
DT	0.246	0.274	0.657	0.641
SVM	0.320	0.289	0.616	0.627
RF	0.387	0.410	0.585	0.564
GBT	**0.400**	**0.406**	**0.575**	**0.562**

From the above tables, we observe that the GBT model performed better than the other four models based on the tradeoff between R squared and RMSE values. We also observe that there is no significant difference in student end-of-term GPA prediction with and without freshman details. This might be due to less sample

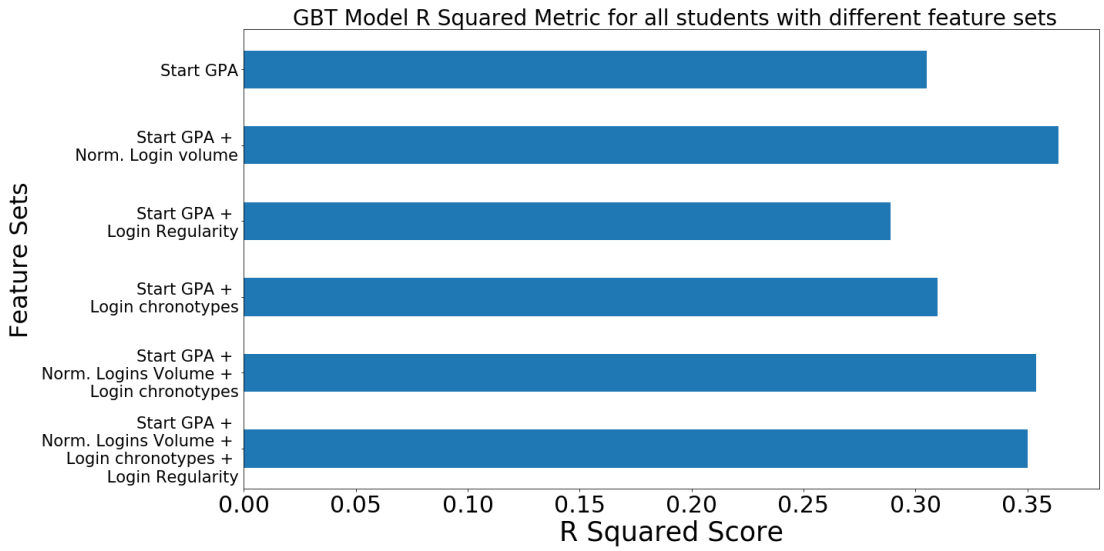


Figure 3. Compare performances of GBT model on different input feature sets.

size (7%) related to freshman cohort. From figure 2, it is also evident that there is a gradual increase in the performance of GBT model as we add data to predictive models as the semester progresses. Even though there is an increase in performance if we add all data captured during the semester, it doesn't help much for real-world interventions as activities that effects student performances will be completed by the end of the semester. Based on this understanding, we focus on data captured until the middle of the semester for feature importance study.

5.2 How do student login and time interval pattern across courses influence student learning outcomes?

To answer research question 2, we adopted a stepwise feature addition study that inputs features by adding one by one into the model and evaluates the performance based on R square and RMSE values. This study is performed on student data collected until the middle of the semester as models developed during this stage will help identify student level indicators and give enough time to deploy interventions that improve student performance. We first start with inputting student Start GPA (Cumulative GPA till the start of Fall 2019 semester) as start GPA showed a high correlation with end-of-term GPA based on our preliminary analysis. We then add normalized login volumes, login regularity, and login chronotypes in a step by step method. Figure 3 shows the R squared performance metric of student-centric models with different input variables.

From figure 3, we observe that students start GPA with normalized student login volumes across courses adds more predictive power to machine learning models. This observation is also supported by earlier studies [39, 40] that showed the importance of student login counts on student course grades and score predictions. Another observation is related to the importance of adding student self-regulation capability based on login regularity measured using entropy statistic. Based on figure 3, we observe that adding login regularity features with student login features and start GPA adds slightly more predictive power compared to model with only login regularity and start GPA features. In addition to these observations, we also observed that login counts based on login chronotypes with start GPA did not add much predictive power to machine learning models. From these results, we also imply that student aggregated login volumes might be adding the same information as login chronotypes.

5.3 Is there a significant variability in feature importance's for students coming from diverse demographics?

One limitation of using the earlier mentioned model-based feature importance study is its inability to explain each feature's importance on different student cohorts. To address this issue and understand the importance of login volumes and regularity features on different student groups, we adopt three approaches: one based on LIME, the second based on correlation analysis, and the third based on linear regression.

5.3.1 LIME based importance's

LIME based approach extract feature importance at the local level, also called local fidelity. By applying the LIME method explained in the methodology section, we extract feature importance's for different student groups categorized based on their demographics.

From figure 4, we can observe that cumulative student GPA at the start of the semester is an important feature to predict student end-of-term GPA. Student login volumes are the second important feature set for model predictions on different student demographics. This study's focus is also on student self-regulation capability measured by the regularity of logins (entropy). We observe that for students with GPA values less than 2, the regularity of logins feature played a key role compared to a student with a higher GPA. This observation also holds for students from minority ethnicity. One implication from these observations This observation suggests that introducing teaching practices that guide LMS use and time management will significantly impact students with low GPA and from a minority race. Start GPA played a slightly less significant role in transfer students than regular students as transfer students join in different years and their cumulative GPA might not be available at the start of the semester, similar to freshman.

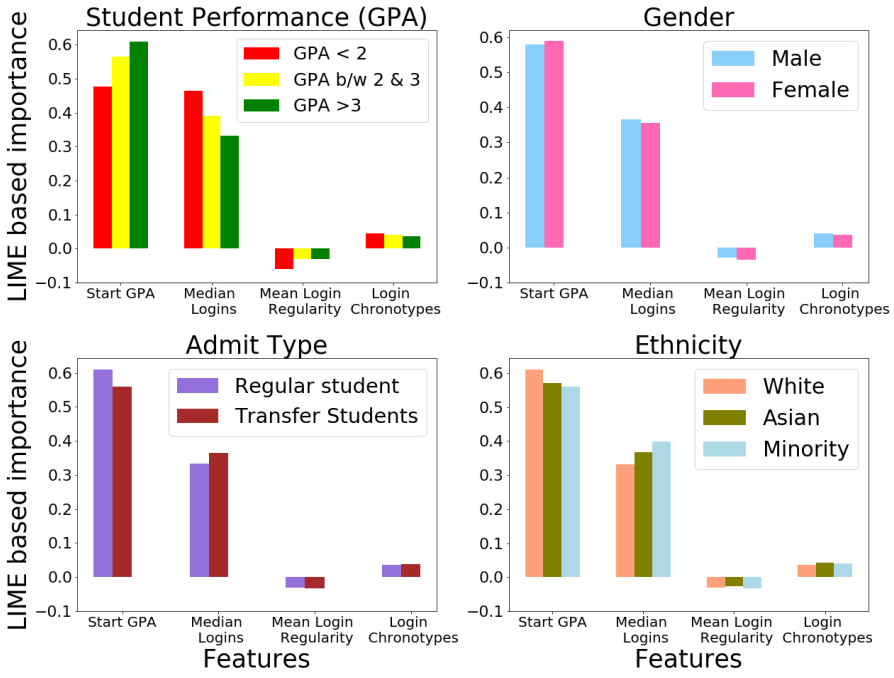


Figure 4. LIME importance's for different student groups divided based on GPA, ethnicity, admit type and gender.

Even though there is a huge imbalance in the number of male and female students present in the dataset, we do not observe any significant difference in feature importance's between these two genders. One limitation of the LIME method is related to global importance's. The importance's showed by LIME at the local level do not necessarily correspond to global importance's. Based on this limitation, we can infer which feature is essential for different students' groups but not quantify them as the importance's calculated in this study are the aggregate of importance's provided by LIME for each individual student.

5.3.2 Correlation based Feature Importance's

As earlier feature importance methods showed a significant impact of login volumes and login regularity measured by entropy statistic to predict student performance, we adopt Pearson correlation statistic to infer this relationship for different student groups. To do this, we create subsets of student data based on different groups: student GPA, gender, ethnicity, and admit type.

From figure 5, we observe that the student logins count and regularity in logins is highly significant for a student with a GPA lower than 2. We can also observe that as the entropy increases, the GPA reduces. This observation holds true as regularity in student logins represents their self-regulation capabilities. Earlier research showed that students with good self-regulation capabilities perform better in class [49, 50]. For other student groups divided based on gender and admit type, there is no significant variation in the importance of logins and entropy on student performances.

Even though the absolute values of correlation observed in figure 5 are not very strong, the comparison between different groups helps understand which features are significant for students from different demographics. In addition to this, we also observe a similar pattern in LIME based importance's discussed in earlier

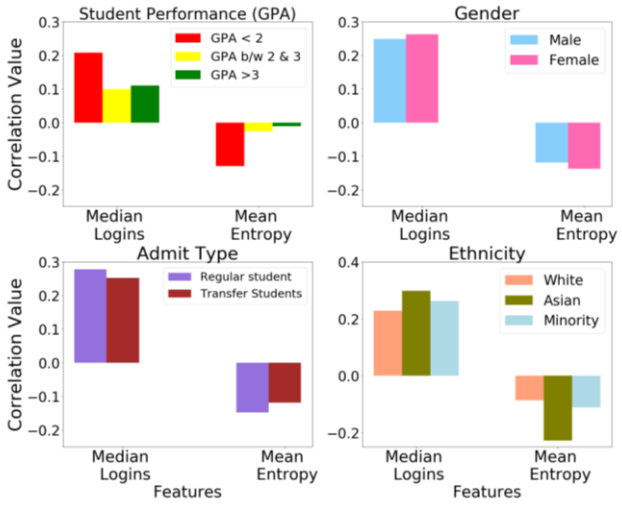


Figure 5. Correlation values for different student groups divided based on GPA, ethnicity, admit type and gender.

sections. We can infer that LIME based method also scales well for global feature importance in this study.

5.3.3 Regression Modeling for Feature Importance

One significant limitation of earlier methods is their inability to capture interaction effects as feature importance might change in the presence of other features. To study the interaction effects, we apply a linear regression model on different categories of student login data collected till the middle of semester. These student categories were divided based on GPA, gender, admit type and ethnicity of students. Even though linear regression models are applied on all features discussed in earlier sections, we only report the coefficients of median login volume and mean login regularity

in table 7, as these variables are the focus of this study. From table 7, we observe that login volumes, and login regularity features are following similar direction for students with lower GPA and students from minority ethnic backgrounds as observed in the LIME and correlation based analysis. There are some discrepancies in other observations as there is no statistical significance (high p values) for coefficients in these cases. Another reason for focusing on student from these two groups is their higher attrition rates found in earlier studies [43, 44]. Studying these groups closely will help develop targeted interventions in the future.

Table 7. Regression coefficients (Significance marked with *)

Student Demographic	Student Groups	Median Logins Coefficient	Mean Login Regularity Coefficient
GPA	**GPA <= 2**	**0.171***	**-0.398***
	GPA >2 & <= 3	-0.013	0.200
	GPA >3	0.065	-0.002
Gender	Male	0.135	0.130
	Female	-0.021	-0.157
Admit Type	Regular	0.399	-0.004
	Transfer	0.611	0.191
Ethnicity	White	0.204	-0.029
	Asian	-0.085	0.201
	Minority Race	**0.201***	**-0.153***

6. DISCUSSION & CONCLUSION

There is a growing interest in building models that capture student behavioral patterns while using LMS systems to predict their performance. Earlier research showed that building efficient models based on LMS data to predict student performances is not a simple task as multiple learning and demographic factors impact student learning processes. Although earlier research in EDM and LA tried to address different issues related to student performance tracking, there is still a gap in developing models that accurately predict overall student performance and explain underlying factors that improve their academic performance. As a step in this direction, this study presents a student-centric modeling approach based on aggregated LMS features to predict and explain the reasons behind varying student performances. This context is both relevant and timely given the increase of LMS adoption and a need for efficient and interpretable model development.

6.1 Key Contributions

One primary contribution in this study is the development of student-centric models on aggregated student LMS login data that are least biased towards the diverse course contents and instructor teaching styles. Using the feature extraction methods developed in this study, we were able to build efficient GBT model that is able to predict student end-of-term GPA with an average R squared of 0.37 across the semester. Furthermore, models built at different durations of a semester showed only slight improvement in predictive performance after crossing a specific duration (middle of the semester). This observation helps develop models in the middle of the semester to estimate student performance accurately.

In addition to developing student-centric models, this study also focused on understanding the impact of various LMS features on student performances. Earlier studies in this domain primarily focused on volume of logins. In this work, we also studied the impact of login regularity measured by entropy statistics on student performance by implementing LIME explanation, correlation, and linear regression methods. From our interpretation studies, we observed that students who login regularly into the LMS system have a positive relationship with performance improvement. This observation is highly significant for underperforming students (GPA < 2) and students from minority races.

We also found no significant difference in the impact of LMS features on Male and Female students. This observation is valid as LMS features used in this study are captured objectively rather than subjectively. This observation also holds for regular and transfer students.

Our study also extracted student interaction features based on concepts in chronobiology and chronopsychology to understand if there is a student performance variation based on different chronotypes. From the results, we observed no significant difference in performance. The impact of these features is negligible in the presence of aggregated student login volume.

6.2 Applications & Limitations

Student performance tracking is a complex process as it depends on multiple dimensions and facets. Developing student-centric models to predict student performance models helps student counselors and educational administrators design student level interventions that attract students' attention. Also, developing predictive models that estimate students' overall performance in the middle of the semester will make them aware of their predicted end-of-term performance. These predictions might act as an external intervention to improve their performance in the remaining part of the semester. By understanding the difference in the impact of LMS features on students from different demographics, researchers and administrators can build more personalized instructional methods that are suitable for diverse student cohorts.

There were also some limitations in this study. The predictive performance achieved by using aggregate features across different courses enrolled by students is moderate at best. It would be more helpful to explore ways to improve the performance of these models. One possibility is to add other features that target independent content access durations, mid-semester assessments, and other external factors. One major challenge that needs to be addressed in our future studies is to find an effective method to aggregate content level features across different courses enrolled by a student. The dataset used in this study is extracted in a single semester and students from two departments that are closely related to each other. To understand if the findings in this study are scalable to other undergraduate students, we will extend these models to students from various departments in the university.

To conclude, we built student-centric models to predict student performances that supports the development of student level interventions. We then use the LIME explanations to study LMS features' importance on student performance prediction. Finally, we study the univariate and multivariate feature importance's using correlation and regression methods and assess them with the feature importance's extracted in LIME method.

7. REFERENCES

[1] Ninoriya, S., Chawan, P. M., & Meshram, B. B. (2011). CMS, LMS and LCMS for elearning. International Journal of Computer Science Issues (IJCSI), 8(2), 644.

[2] Romero, C., & Ventura, S. (2007). Educational data mining: A survey from 1995 to 2005. Expert systems with applications, 33(1), 135-146.

[3] Plak, S., Cornelisz, I., Meeter, M., & Van Klaveren, C. (2019, March). Early Warning Systems for More Effective Student Counseling in Higher Education–Evidence from a Dutch Field Experiment. In Proceedings of the SREE Spring 2019 Conference, Washington, DC, USA (pp. 6-9).

[4] Beck, H. P., & Davidson, W. D. (2001). Establishing an early warning system: Predicting low grades in college students from survey of academic orientations scores. Research in Higher education, 42(6), 709-723.

[5] Conijn, R., Snijders, C., Kleingeld, A., & Matzat, U. (2016). Predicting student performance from LMS data: A comparison of 17 blended courses using Moodle LMS. IEEE Transactions on Learning Technologies, 10(1), 17-29.

[6] Mozahem, N. A. (2020). Using learning management system activity data to predict student performance in face-to-face courses. International Journal of Mobile and Blended Learning (IJMBL), 12(3), 20-31.

[7] Riestra-González, M., del Puerto Paule-Ruíz, M., & Ortin, F. (2021). Massive LMS log data analysis for the early prediction of course-agnostic student performance. Computers & Education, 163, 104108.

[8] Chiu, Y. C., Hsu, H. J., Wu, J., & Yang, D. L. (2018). Predicting Student Performance in MOOCs Using Learning Activity Data. J. Inf. Sci. Eng., 34(5), 1223-1235.

[9] Alokluk, J. A. (2018). The Effectiveness of Blackboard System, Uses and Limitations in Information Management. Intelligent Information Management, 10(06), 133.

[10] Berechet, L. D., & Georgescu, M. The Road from Blackboard Learning Management System to Moodle Learning Management System in Modern Universities.

[11] Shchedrina, E., Valiev, I., Sabirova, F., & Babaskin, D. (2021). Providing Adaptivity in Moodle LMS Courses. International Journal of Emerging Technologies in Learning (iJET), 16(2), 95-107.

[12] Ghilay, Y. (2019). Effectiveness of learning management systems in higher education: Views of Lecturers with different levels of activity in LMSs. Ghilay, Y.(2019). Effectiveness of Learning Management Systems in Higher Education: Views of Lecturers with Different Levels of Activity in LMSs. Journal of Online Higher Education, 3(2), 29-50.

[13] Aldiab, A., Chowdhury, H., Kootsookos, A., Alam, F., & Allhibi, H. (2019). Utilization of Learning Management Systems (LMSs) in higher education system: A case review for Saudi Arabia. Energy Procedia, 160, 731-737.

[14] Viberg, O., Hatakka, M., Bälter, O., & Mavroudi, A. (2018). The current landscape of learning analytics in higher education. Computers in Human Behavior, 89, 98-110.

[15] Lang, C., Siemens, G., Wise, A., & Gasevic, D. (Eds.). (2017). Handbook of learning analytics. New York, NY, USA: SOLAR, Society for Learning Analytics and Research.

[16] Shum, S. B., & Crick, R. D. (2012, April). Learning dispositions and transferable competencies: pedagogy, modelling and learning analytics. In Proceedings of the 2nd international conference on learning analytics and knowledge (pp. 92-101).

[17] Shum, S. B., Ferguson, R., & Martinez-Maldonado, R. (2019). Human-centred learning analytics. Journal of Learning Analytics, 6(2), 1-9.

[18] Dawson, S., Heathcote, L., & Poole, G. (2010). Harnessing ICT potential. International Journal of Educational Management.

[19] Vengroff, R., & Bourbeau, J. (2006, February). In-class vs. On-line and Hybrid Class Participation and Outcomes: Teaching the Introduction to Comparative Politics Class. In annual meeting of the APSA Teaching and Learning Conference.

[20] Dutt, A., & Ismail, M. A. (2019, June). Can we predict student learning performance from LMS data? A classification approach. In 3rd International Conference on Current Issues in Education (ICCIE 2018) (pp. 24-29). Atlantis Press.

[21] Lust, G., Vandewaetere, M., Ceulemans, E., Elen, J., & Clarebout, G. (2011). Tool-use in a blended undergraduate course: In Search of user profiles. Computers & Education, 57(3), 2135-2144.

[22] Hung, J. L., & Zhang, K. (2012). Examining mobile learning trends 2003–2008: A categorical meta-trend analysis using text mining techniques. Journal of Computing in Higher education, 24(1), 1-17.

[23] Dawson, S., Heathcote, L., & Poole, G. (2010). Harnessing ICT potential. International Journal of Educational Management.

[24] Damianov, D. S., Kupczynski, L., Calafiore, P., Damianova, E. P., Soydemir, G., & Gonzalez, E. (2009). Time spent online and student performance in online business courses: A multinomial logit analysis. Journal of Economics and Finance Education, 8(2), 11-22.

[25] Baugher Varanelli Weisbord, D. A. E. (2003). Student hits in an internet-supported course: How can instructors use them and what do they mean?. Decision Sciences Journal of Innovative Education, 1(2), 159-179.

[26] Biktimirov, E. N., & Klassen, K. J. (2008). Relationship between use of online support materials and student performance in an introductory finance course. journal of education for business, 83(3), 153-158.

[27] Coldwell, J., Craig, A., Paterson, T., & Mustard, J. (2008). Online students: Relationships between participation, demographics and academic performance. Electronic journal of e-learning, 6(1), 19-30.

[28] Greenland, S. J. (2011). Using log data to investigate the impact of (a) synchronous learning tools on LMS interaction. In Proceedings of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE) Conference.

[29] Nandi, D., Hamilton, M., Harland, J., & Warburton, G. (2011, January). How active are students in online discussion forums?. In Proceedings of the Thirteenth Australasian Computing Education Conference-Volume 114 (pp. 125-134).

[30] Shum, S. B., & Ferguson, R. (2012). Social learning analytics. Journal of educational technology & society, 15(3), 3-26.

[31] Tempelaar, D. T., Rienties, B., & Giesbers, B. (2015). In search for the most informative data for feedback generation: Learning analytics in a data-rich context. Computers in Human Behavior, 47, 157-167.

[32] Agudo-Peregrina, Á. F., Iglesias-Pradas, S., Conde-González, M. Á., & Hernández-García, Á. (2014). Can we predict success from log data in VLEs? Classification of interactions for learning analytics and their relation with performance in VLE-supported F2F and online learning. Computers in human behavior, 31, 542-550.

[33] Dominguez, M., Bernacki, M. L., & Uesbeck, P. M. (2016). Predicting STEM Achievement with Learning Management System Data: Prediction Modeling and a Test of an Early Warning System. In EDM (pp. 589-590).

[34] Lerche, T., & Kiel, E. (2018). Predicting student achievement in learning management systems by log data analysis. Computers in Human Behavior, 89, 367-372.

[35] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

[36] Hofmann, M., & Klinkenberg, R. (Eds.). (2016). RapidMiner: Data mining use cases and business analytics applications. CRC Press.

[37] Mandalapu, V., & Gong, J. (2019, January). Studying Factors Influencing the Prediction of Student STEM and Non-STEM Career Choice. In Proceedings of the 12th International Conference on Educational Data Mining.

[38] Mandalapu, V., Gong, J., & Chen, L. (2021). Do we need to go Deep? Knowledge Tracing with Big Data. arXiv preprint arXiv:2101.08349.

[39] Kennedy, G., Coffrin, C., De Barba, P., & Corrin, L. (2015, March). Predicting success: how learners' prior knowledge, skills and activities predict MOOC performance. In Proceedings of the fifth international conference on learning analytics and knowledge (pp. 136-140).

[40] Stapel, M., Zheng, Z., & Pinkwart, N. (2016). An Ensemble Method to Predict Student Performance in an Online Math Learning Environment. International Educational Data Mining Society.

[41] Putilov, A. A., Sveshnikov, D. S., Puchkova, A. N., Dorokhov, V. B., Bakaeva, Z. B., Yakunina, E. B., ... & Mairesse, O. (2021). Single-Item Chronotyping (SIC), a method to self-assess diurnal types by using 6 simple charts. Personality and Individual Differences, 168, 110353.

[42] Romo-Nava, F., Blom, T. J., Guerdjikova, A., Winham, S. J., Cuellar-Barboza, A. B., Nunez, N. A., ... & McElroy, S. L. (2020). Evening chronotype, disordered eating behavior, and poor dietary habits in bipolar disorder. Acta Psychiatrica Scandinavica, 142(1), 58-65.

[43] Kupczynski, L., Gibson, A. M., Ice, P., Richardson, J., & Challoo, L. (2011). The impact of frequency on achievement in online courses: A study from a south Texas university. Journal of Interactive Online Learning, 10(3).

[44] Liu, F., & Cavanaugh, C. (2011). High enrollment course success factors in virtual school: Factors influencing student academic achievement. International Journal on E-learning, 10(4), 393-418.

[45] Kozachenko, L. F., & Leonenko, N. N. (1987). Sample estimate of the entropy of a random vector. Problemy Peredachi Informatsii, 23(2), 9-16.

[46] Kraskov, A., Stögbauer, H., & Grassberger, P. (2004). Estimating mutual information. Physical review E, 69(6), 066138.

[47] Ruping, S. (2000). mySVM-manual. http://www-ai. cs. uni-dortmund. de/SOFTWARE/MYSVM/.

[48] Gray, R. M. (2011). Entropy and information theory. Springer Science & Business Media.

[49] Kim, J. H., Seodaemun-gu, S., Park, Y., Song, J., & Jo, I. H. (2014). Predicting students' learning performance by using online behavior patterns in blended learning environments: comparison of two cases on linear and non-linear model. Korea, 120, 750.

[50] Park, Y., & Jo, I. H. (2015). Development of the learning analytics dashboard to support students' learning performance. Journal of Universal Computer Science, 21(1), 110.

[51] Jo, I. H., Kim, D., & Yoon, M. (2015). Constructing proxy variables to measure adult learners' time management strategies in LMS. Journal of Educational Technology & Society, 18(3), 214-225.

[52] Jo, I. H., Park, Y., Yoon, M., & Sung, H. (2016). Evaluation of online log variables that estimate learners' time management in a Korean online learning context. International Review of Research in Open and Distributed Learning, 17(1), 195-213.

Generative Grading: Near Human-level Accuracy for Automated Feedback on Richly Structured Problems

Ali Malik[1*], Mike Wu[1*],
Vrinda Vasavada[1], Jinpeng Song[1], Madison Coots[1],
John Mitchell[1], Noah Goodman[1,2], Chris Piech[1]

[1]Department of Computer Science, Stanford University
[2]Department of Psychology, Stanford University
{malikali, wumike, vrindav, jsong5, mcoots, jcm, ngoodman, piech}@cs.stanford.edu

ABSTRACT

Access to high-quality education at scale is limited by the difficulty of providing student feedback on open-ended assignments in structured domains like programming, graphics, and short response questions. This problem has proven to be exceptionally difficult: for humans, it requires large amounts of manual work, and for computers, until recently, achieving anything near human-level accuracy has been unattainable. In this paper, we present generative grading: a novel computational approach for providing feedback at scale that is capable of accurately grading student work and providing nuanced, interpretable feedback. Our approach uses generative descriptions of student cognition, written as probabilistic programs, to synthesise millions of labelled example solutions to a problem; we then learn to infer feedback for real student solutions based on this cognitive model.

We apply our methods to three settings. In block-based coding, we achieve a 50% improvement upon the previous best results for feedback, exceeding human-level accuracy. In two other widely different domains—graphical tasks and short text answers—we achieve improvements over the previous state of the art by about 4x and 1.5x respectively, approaching human accuracy. In a real classroom, we ran an experiment with our system to augment human graders, yielding doubled grading accuracy while halving grading time.

Keywords

Generative models, automated feedback, Idea2Text, probabilistic programs, grammars, Zipf distribution, zero shot.

1. INTRODUCTION

Enabling global access to high-quality education is a long-standing challenge. The combined effect of increasing costs per student [3] and rising demand for higher education makes this issue particularly pressing. A major barrier to providing quality education has been the ability to automatically provide *meaningful and accurate* feedback on student work.

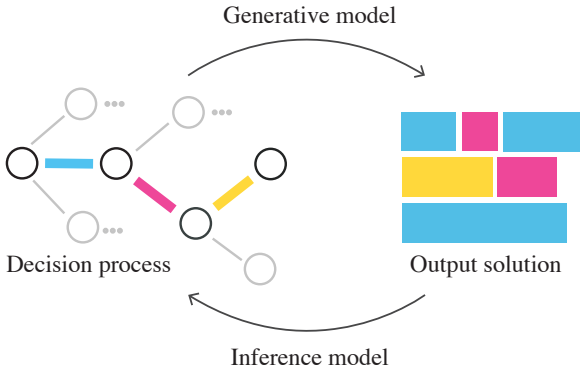


Figure 1: Students solve problems by making decisions that result in their final solution (generative model). Providing feedback requires the reverse task of seeing a solution and inferring the student decisions that lead to this solution (inference model).

Learning to provide feedback on richly structured problems beyond simple multiple-choice has proven to be a hard machine learning problem. Five issues have emerged, many of which are typical of human-centred AI problems: (1) student work is extremely diverse, exhibiting a heavy tailed distribution where most solutions are rare, (2) student work is difficult and expensive to label with fine-grained feedback, (3) we want to provide feedback (without historical data) for even the very first student, (4) grading is a precision-critical domain with a high cost to misgrading, and (5) predictions must be explainable and justifiable to instructors and students. Despite extensive research using massive education data [23, 1, 33, 28, 18, 14], these issues make traditional supervised learning inadequate for automatic feedback.

Human instructors are experts at providing feedback. When grading assignments, they have an understanding of the decisions and missteps students might make when solving a problem, and what corresponding solutions these choices would result in. For example, an instructor understands that a student wanting to repeat something in a programming assignment might use a `for` loop or manually write out repeated statements. And given that the student uses a loop, their loop could be correct or off-by-one.

Ali Malik, Mike Wu, Vrinda Vasavada, Jinpeng Song, Madison Coots, John Mitchell, Noah Goodman and Chris Piech "Generative Grading: Near Human-level Accuracy for Automated Feedback on Richly Structured Problems". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 275-286. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

In essence, instructors mentally possess *generative models* of student decision-making and how these decisions manifest in a final solution (Fig. 1, forwards). When providing feedback, the instructor does *inference*: given a student solution, they use their mental model to try and determine the underlying student decisions that could have resulted in this solution (Fig. 1, backwards).

In this paper, we propose an automated feedback system that mimics the instructor process as closely as possible. Firstly, the system elicits a literal generative model from an instructor in the form of a concrete student simulator. Secondly, it uses deep neural networks and a novel inference method with this simulator to learn how to do inference on student solutions, *without using any labelled data*. Finally, the inference model is used to provide automated feedback to real student solutions. We call this end-to-end approach **generative grading**.

When used across a spectrum of public education data sets, our automated feedback system is able to grade student work with close to *expert human-level* fidelity. In block-based coding, we exceed human-level accuracy, achieving a 50% improvement over the previous best results for feedback. In two other widely different domains—graphical tasks and short text answers—we achieve improvements over the previous state of the art by about 4x and 1.5x respectively, approaching human accuracy. We used our system in a real classroom to augment human graders in a CS1 class, yielding *doubled* grading accuracy while *halving* grading time.

1.1 Main contributions

In Sec. 4, we present an easy-to-use and highly expressive class of generative models called Idea2Text simulators that allow an instructor to encode their mental models of student decision-making. These simulators can succinctly express student decisions and how these decisions manifest in a final solution for a broad set of problem domains like graphics programming, short-answer questions, and introductory programming in Java. We provide a Python implementation that allows any instructor to easily write these simulators.[1]

In Sec. 5, we show how to use Idea2Text simulators with deep neural networks to infer students' decision processes from their solutions. This extracted decision process is a general representation of a student's solution and can be used for several downstream tasks such as providing automated feedback, assisting human grading, auditing and interpreting the model decisions, and improving the quality of the simulator itself.

In order to do inference successfully on our expressive class of simulators, we must overcome several interesting technical challenges (Sec. 5). Learning to map solutions to sequences of decisions specified by the simulator is a nonstandard machine learning task, with non-fixed labels, varied sequence lengths, and unexpected trajectories. Moreover, generating simulated training data from the simulators requires an intelligent sampling method to work effectively.

[1]All code publicly available at: `https://github.com/malik-ali/generative-grading`

In Sec. 6 we show the efficacy of our approach in practice on a diverse set of richly structured problems. We attain close to human-level accuracy on providing feedback and surpass many previous state of the art results. We also discuss several interesting extensions in Sec. 7 that use our system to go beyond just providing automated feedback.

The generative grading system is powerful because it addresses many of the issues of traditional supervised learning mentioned above. We find that the cost of writing simulators for a new assignment is orders of magnitude cheaper for instructors than manually annotating individual student work. The simulators allow us to sample infinite data, and our adaptive sampling strategy lets us explore diverse student solutions in our training data. It is "zero-shot", requiring no historical data nor annotation, and thus works for the very first student. Moreover, our novel inference system allows for interpretable and explainable decisions.

2. RELATED WORK

"Rubric sampling" [32] first introduced the concept of encoding expert priors in grammars of student decisions, and was the inspiration for our work. The authors design Probabilistic Context Free Grammars (PCFGs) to curate synthetically labelled datasets to train supervised classifiers for feedback. Our approach builds on this, but presents a more expressive family of generative models of student decision-making that are context sensitive and comes with new innovations that enable effective inference. From our results on Code.org, we see that this expressivity is responsible for significant improvements in our model's performance. Furthermore, this prior work only used to PCFGs to create simulated datasets of feedback labels for supervised learning. In contrast, we learn to infer the entire decision trajectory of a student solution, allowing us to do things like dense feedback and human-in-the-loop grading.

We draw theoretical inspiration for our generative grading system from Brown's "Repair Theory" which argues that the best way to help students is to understand the generative origins of their mistakes [4]. Building systems of student cognition has been used in K-12 arithmetic problems [16] and subtraction mistakes [8].

Automated feedback for open-ended richly structured problems has been studied through a few lenses. In many approaches, traditional supervised learning is employed to map solutions to feedback [21, 30, 1, 33]. These methods require large hand-labelled datasets of diverse student solutions, which is difficult due to heavy-tailed distributions. Feedback specific to computer programming problems has been explored based on executing student solutions and comparing to a reference solution [13, 19]. An interesting parallel to our work is found in [13], where the instructor is asked to specify the kinds of mistakes students can make. These approaches are limited to code and don't provide feedback on the problem-solving process of a student.

Extracting expert-written generative models for inference has seen enormous use in fields where domain expertise is critical. Some key example include medical diagnosis, engineering, ecology, and finance, where a generative model like a Bayesian network is elicited from experts. [20, 22]. In ed-

ucation, instructors have domain expertise about students, and Idea2Text serves as an easy-to-use generative model for instructors to encode this expertise.

Inference over decision trajectories of Idea2Text simulators is similar to "compiled inference" for execution traces in probabilistic programs. As such, our inference engine shares similarities to this literature [25, 17]. With Idea2Text simulators, we get a nice interpretation of compiled inference as a parsing algorithm.

3. BACKGROUND

In this section, we introduce the feedback challenge and what makes it a difficult machine learning problem.

3.1 Feedback as a Prediction Problem

The *feedback prediction* task is to automatically provide feedback to a given student solution. While this is easy to do for simple multiple-choice problems, we focus on the challenging task of providing feedback on richly-structured problems like computer programming or short-answer responses.

Both the type of student solutions and the type of feedback required for the task can take many forms. A student solution can be a piece of text, which could represent problems like an essay, a maths proof, or a code snippet. It could also be graphical output in the form of an image. Similarly, feedback can take the form of something simple like classifying solutions to a fixed set of misconceptions, or something complex such as highlighting and annotating specific parts of a student solution

3.2 Difficulty of Automated Feedback

Feedback prediction on richly structured problems has been an extremely difficult challenge in education research. Even limited to simple problems in computer science like beginner block-based programming, automated solutions to providing feedback have been restricted by scarce data and lack of robustness. We discuss a few of the properties of student work that make predicting feedback such a difficult challenge.

(1) Heavy-tailed Distributions: Student work in the form of natural language, mathematical symbols, or code follow heavy-tailed Zipf distributions. This means that a few solutions are extremely common whereas almost all other examples are unique and show up rarely. Fig. 2 plots the log-frequency of unique examples against the log of the rank across four datasets of student work in block-based programming code, Java code, and free response. For all datasets, we observe a linear relationship in log-log space, which is a characteristic property of Zipf distributions.

These heavy-tailed Zipf distributions pose a hard generalisation problem for traditional supervised machine learning: a handful of similar examples appear very frequently in the training data whereas almost all other examples are unique. This means at test time, examples are likely to introduce unseen tokens, new misconceptions, and novel student strategies. In a Zipf distribution, even if we observe a million student solutions, there is roughly a 15% chance that the next student generates a unique solution.

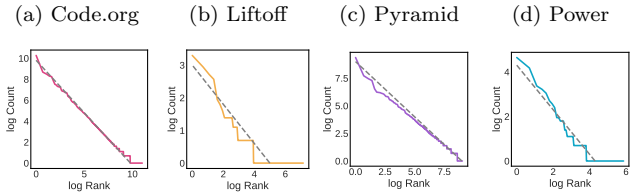


Figure 2: Student solutions (across many domains) exhibit heavy-tailed Zipf distributions, meaning a few solutions are extremely common but all other solutions are highly varied and show up rarely. This suggests that the probability of a student submission not being present in a dataset is high, making supervised learning on a small data set difficult.

(2) Difficulty of Annotation: Annotating student work with feedback requires an instructor-level expertise of the domain. Providing fine-grained feedback also takes effort to read and understand student solutions before inferring possible misconceptions. In [32], the authors found that it took 26 hours to label 800 student solutions to block-based programming.

This difficulty, combined with Zipf properties, makes supervised learning challenging. As an example, in 2014, Code.org, a widely used resource for beginners in computer science, ran an initiative to crowdsource thousands of instructors to label 55,000 student solutions in a block-based programming language. Yet, despite having access to an unprecedented amount of labelled data, traditional supervised methods failed to perform well on even these "simple" questions.

(3) Limitations on Data Size: Even if the Code.org approach succeeded, most classrooms do not share the same scale as Code.org. A method that relies heavily on historical data is not widely applicable in the average classroom setting. In our experiments, our data sets contain less than a few hundred examples, again disqualifying the application of supervised algorithms. The ideal feedback model will be zero-shot so that it works even for the very first student.

4. MODELLING STUDENT COGNITION

Having presented the feedback challenge and motivated why supervised learning cannot solve this problem alone, we discuss the idea of modelling the cognitive process of student decision-making when producing a solution.

When an instructor provides feedback on student work, they often have a latent mental model of student decision making; this captures the kinds of steps and mistakes they think students will make and what solutions are indicative of those steps. The instructor (or TAs) then grade solutions, one at a time, by essentially inferring the steps in the decision-making model that lead to the produced solution.

As a concrete example from introductory programming, suppose a student is trying to print a countdown of numbers from 10 to 1. An instructor understands that as a first step, a student might use a `for` loop or manually write ten `print` statements. Given that the student uses a loop, they could increment up or down. And given that they increment down, their loop comparison could be correct or off-by-one. At each of these decision points, the instructor can conceive how a specific choice would manifest in the solution.

We want to allow instructors to capture their mental model of student decision-making for a given problem and distil it into a concrete executable simulator that generates solutions. Such a generative model could be used to simulate unlimited student data, including their decision process and resulting solutions. This simulated dataset can then be used for learning how to provide feedback.

In this section, we formalise the idea of a student's decision process for generating the solution to a problem and discuss how we can represent the instructor's latent model of this generative process as a concrete simulator.

4.1 Student Decision Process (SDP)

A student's decision process (SDP) can be seen as the sequence of choices the student makes while solving a problem, and how those choices correspond to different outputs. The sequence is intended to reflect all of the critical decisions made by the student, and in particular those a teacher would like to recover from the solution. Importantly, not all students will encounter the same sequence of decisions since an early decision choice may determine which decisions are faced later in problem solving.

We can formally think of a decision point as a categorical random variable, X, and the specific choices that can be made as the different values, $x \in Val(X)$, that the random variable can take. The decision process of a student can be seen as a sequence trajectory $(X_t, x_t)_{t=1}^{T}$, of decisions encountered and made by the student in solving the problem.

Under this interpretation, specifying a model for an SDP amounts to defining the space of possible decision trajectories and a probability distribution over this space. By the chain rule for probabilities, we can decompose this overall distribution as a sequence of conditional probabilities $\mathbb{P}(X_t = x_t \mid \mathbf{X}_{<t} = \mathbf{x}_{<t})$. Here x_t is the choice made at step t from domain X_t and $\mathbf{X}_{<t}$ is shorthand for the sequence of decisions before time t.

We want to make this general formulation more tractable to allow us to specify useful SDPs as generative models we can sample from. Prior work [32] has attempted to express SDPs by restricting the class of generative models to probabilistic context free grammars (PCFGs). They found that instructor-written PCFGs could often be used to emulate student problem-solving and generate student solutions for small problems. In this setting, the non-terminal nodes of the PCFG represent decisions to be made (e.g. syntax confusion) and the production rules represent how decisions are made and manifested into output (e.g. the code is missing a semicolon). Instructors create student simulators by specifying decision points, rules, and probabilities for each rule (e.g. missing a semicolon is much more likely than missing a `function` statement).

A PCFG is compact and useful, but makes the independence assumption that that the choice made at time t is independent of past choices made while solving the problem i.e. $\mathbb{P}(X_t = x_t \mid \mathbf{X}_{<t} = \mathbf{x}_{<t}) = \mathbb{P}(X_t = x_t)$. This context-independence is a strong restriction that severely limits what instructors can express about the student decision-making process and fails to faithfully model student reasoning. As can be seen in even the simple countdown example above, the off-by-one error would manifest differently in student output depending on whether the student chose to increment up or down. Thus, context dependence of decision making is an important property to model.

Algorithm 1 Idea2Text Simulation

Input: Idea2Text simulator (D, Σ, S)
Output: Tuple (τ, y) of decision trajectory and output solution.

procedure SIMULATE(D, Σ, S)
  $\tau \leftarrow [\,]$
  $y \leftarrow$ GENERATE(S, τ) ▷ Begin from start node
  return (τ, y)
procedure GENERATE(N, τ)
  $a, X_a, \Pi_a \leftarrow N$ ▷ Unpack current decision node
  $x_a, y \leftarrow \Pi_a(X_a, \tau)$ ▷ Get decision choice and output
  $\tau.append((a, x_a))$
  for decision node N' in y **do** ▷ In order left to right
    $y' \leftarrow$ GENERATE(N', τ)
    $y \leftarrow$ REPLACE(y, N', y') ▷ Replace N' with y'
  return y

4.2 Idea2Text

In this section, we define a broader class of generative models that is powerful enough to capture more complexities of expert models of student cognition. Similar to PCFGs we structure our models around a set of non-terminal symbols that correspond to student decisions and contribute to the final output. However, drawing from work on probabilistic programs [10, 11], we allow these choices to be made depending on previous choices. While dependence on context leads to extremely expressive models, we will show that requiring some text to be generated at each step is enough for inference to remain tractable. We call this class of generative models Idea2Text.[2]

Concretely, an Idea2Text simulator consists of a tuple of (D, Σ, S) denoting a set of nonterminal decision nodes, a set of terminal nodes, and a starting root node, respectively. Intuitively, decision nodes correspond to decisions a student might make and the terminal nodes correspond to literal text tokens in the final output. Each run of the simulator also keeps a global state τ which stores the history of all decisions made during the execution (often called an *execution trace* for probabilistic programs [2, 29]).

Each decision node in D is a tuple (a, X_a, Π_a) consisting of a unique name, a random variable representing the decision choices, and a production program, which (1) specifies how this decision should be made based on the decisions made so far, and (2) produces an output solution for a given decision choice.

More concretely, the production program is a probabilistic function that takes the current decision history τ and does the following:

(1) Samples the random variable $x_a \sim X_a$, from a distribution, $\mathbb{P}(X_a|\tau)$, that can depend on the decision history.

[2] A very similar class of models, used in the very different domain of customer service, was independently named Idea2Text by scientists at Gamalon, Inc.

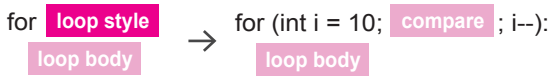


Figure 3: An example decision expansion step in an Idea2Text simulation. The "loop style" decision node chooses a type of for loop (e.g. increment vs decrement) and outputs a string containing the header of the for loop (terminals) plus another decision node.

(2) Based on the sampled choice, x_a, produces an output string, y, consisting of literal text (terminal nodes) and incomplete segments (decision nodes). These incomplete segments correspond to future decisions that will be expanded later (see Fig. 3).

(3) Returns the sampled choice and output string: (x_a, y).

An output from Idea2Text is a generation from the root node to a sequence of literal text by recursively expanding the decision nodes, as shown in Algorithm 1. Each output is associated with the final decision trajectory, $\tau = [(a_t, x_{a_t})]_{t=1}^{T}$, of random variables encountered during generation. Here, a_t denotes the unique name for the random variable encountered at timestep t, and x_{a_t} is the sampled value for that variable.

We point out some important properties of Idea2Text simulators. First, each decision node's choice can depend on the decision history τ, allowing past decisions to influence current behaviour. This is strictly more expressive than PCFGs [32] and allows instructors to write highly contextual models of student problem-solving. Second, production programs have the full power of programming languages at their disposal and can use arbitrarily complicated transformations to produce their output sequence. As an example, a production program can transform terminal nodes into images or use a machine-learning conjugator to conjugate it's produced text into a proper sentence.

5. INFERENCE

In this section we describe how we can use an instructor-written Idea2Text simulator to learn how to infer the decisions underlying a student solution.

At a high-level, the Idea2Text simulator contains the instructor's mental model for the sequence of decisions students make that result in different solutions. For inference we want to do the reverse: given a student solution, we want to find a trajectory of decisions in the simulator that would produce that solution.

A model that could successfully do this inference could be used to map real student solutions to decision steps in the simulator. These extracted decisions are a rich and general representation of a student's solution and can be used for downstream tasks such as automated feedback, assisting human grading, auditing and interpreting the model decisions, and improving the quality of the simulator.

More formally, let $\mathcal{G}$ be a given Idea2Text simulator. Each execution of $\mathcal{G}$ produces a decision trajectory τ and corresponding production y. Since the execution is probabilistic, the simulator induces a probability distribution $p_{\mathcal{G}}(\tau, y)$ over trajectories and productions.

Given a student solution, y, we are interested in the task of *parsing*: this is the task of mapping y to the most likely trajectory in the Idea2Text simulator, $\arg\max_\tau p_{\mathcal{G}}(\tau|y)$, that could have produced y. This is a difficult search problem: the number of trajectories grows exponentially even for simple grammars, and common methods for parsing by dynamic programming (Viterbi, CYK) are not applicable in the presence of context-sensitivity and functional transformations. What's more, in order to transfer robustly to real student solutions, we would like to be able to approximately parse solutions that are not possible to generate from the simulator, but are sufficiently "nearby".

At a high level, our approach is to construct a large data set from the simulator and then learn an inverse "inference" neural net that can reconstruct the decision trajectory from the solution.

5.1 Adaptive Grammar Sampling

To train our models, we generate a large dataset of N trajectories and their associated productions, $\mathcal{D} = \{(\tau^{(m)}, y^{(m)})\}_{m=1}^{N}$, by repeatedly executing $\mathcal{G}$.

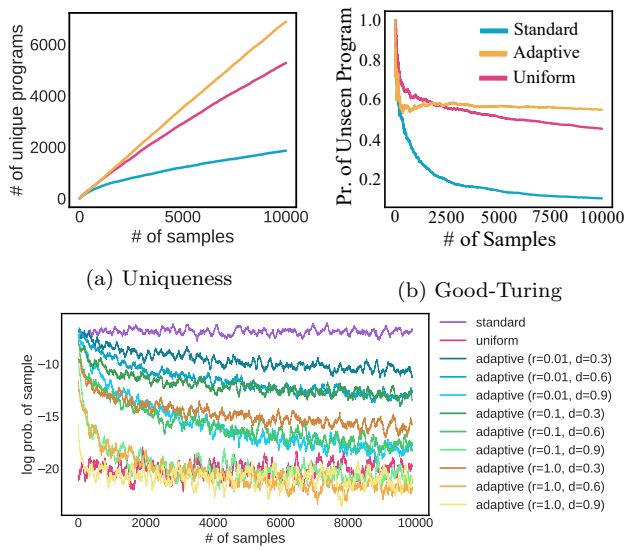
However, due to the Zipf-like nature of student work (see Sec. 3.2), standard i.i.d. sampling from the simulator will tend to over-represent the most probable productions. For our models, the more diverse student cognition we can simulate in the training data, the more we expect to generalise to the long tail of real students. Thus, we need sampling strategies that prioritise diversity.

A simple but flawed idea for generating diverse solutions would be to make choices at decision nodes uniformly randomly instead of using the expert-written distributions. This approach will generate more unique productions, but disregarding the expert-written distributions will result in unlikely and less realistic productions.

Ideally, we want to sample in a manner that covers all the most likely productions first, and then smoothly transition into sampling increasingly unlikely productions. This would generate unique productions efficiently while also retaining the expert-written distributions specified in the production programs. With these desiderata in mind, we propose a method called *Adaptive Grammar Sampling*. For each decision node in the simulator, we down-weight the probability of sampling each choice proportional to how many times it has been sampled in the past. To avoid overly punishing decision nodes early in the execution trace, we discount this down-weighting by a decay factor d that depends on the depth of the decision in the trajectory.[3] This method is inspired by Monte-Carlo Tree Search [5] and shares similarities with Wang-Landau sampling from statistical physics [27].

Fig. 4 shows a comparison of the effectiveness of adaptive sampling to uniform and i.i.d. sampling. Adaptive sampling interpolates nicely between sampling likely examples early on, as i.i.d. sampling does, to sampling unlikely examples

[3]The details can be found in the code.



(a) Uniqueness

(b) Good-Turing

(c) Likelihood of samples over time

Figure 4: Efficiency of sampling strategies for the Liftoff simulator. (a) Number of unique samples vs total samples so far. (b) Probability of sampling a unique next program given samples so far. (c) Likelihood of generated samples over time for different sampling strategies.

later, as uniform-choice sampling does. Note that adaptive sampling is customisable: as shown in 4c, the algorithm has parameters (r and d) that can be adjusted to control how fast we explore increasingly unlikely productions.

5.2 Neural Approximate Parsing

With good diverse samples available, we now aim to learn an approximation to the posterior $p_{\mathcal{G}}(\tau|y)$. We will do so by training a deep neural network to reconstruct the trajectory step-by-step. We call this approach *neural approximate parsing* with generative grading, or GG-NAP.

The challenge of inference over trajectories is a difficult one. Trajectories can vary in length and contain decision nodes with different support. To approach this, we decompose the inference task into a set of easier sub-tasks, similar to [25, 17]. The posterior distribution over a trajectory $\tau = (a_t, x_{a_t})_{t=1}^{T}$ given a production y can be written as the product of individual posteriors over each decision node x_{a_t} using the chain rule:

$$p_{\mathcal{G}}(x_{a_1}, \dots x_{a_T}|y) = \prod_{t=1}^{T} p_{\mathcal{G}}(x_{a_t}|y, \mathbf{x}_{<a_t}) \quad (1)$$

where $\mathbf{x}_{<a_t}$ denotes previous (possibly non-contiguous) nonterminals $(x_{a_1}, \dots, x_{a_{t-1}})$. Eqn. 1 shows that we can learn each posterior $p(x_{a_t}|\mathbf{x}_{<a_t}, y)$ separately. With an autoregressive model $\mathcal{M}$, we can efficiently represent the influence of previous nonterminals $\mathbf{x}_{<a_t}$ using a shared hidden representation over T timesteps. Since most standard choices for $\mathcal{M}$ (e.g. an RNN) require fixed-dimension inputs, we need to encode the solution and the history of choices into consistent vectors.

Firstly, to encode the solution y, we use standard machinery (e.g. CNNs for images, RNNs for text) with a fixed output dimension. To represent the nonterminal choices with different support, we define three layers for each random variable x_{a_t}: (1) a one-hot embedding layer that uses the unique name a_t to lexically identify the random variable, (2) a value embedding layer that maps the value of x_{a_t} to a fixed dimension vector and (3) a value decoding layer that transforms the hidden output state of $\mathcal{M}$ into parameters of the posterior for the next nonterminal x_{a_t+1}. Thus, the input to the $\mathcal{M}$ is a fixed size, being the concatenation of the value embedding, name embedding, and production encoding.[4]

To train the GG-NAP, we optimise the objective,

$$\mathcal{L}(\theta) = \mathbb{E}_{p_{\mathcal{G}}(\tau,y)}[\log p_\theta(\tau|y)] \approx \frac{1}{M}\sum_{m=1}^{N} \log p_\theta(\tau^{(m)}|y^{(m)}) \quad (2)$$

where θ are all trainable parameters and $p_\theta(\tau|y)$ represents the posterior distribution defined by the inference engine. At test time, given only a production y, GG-NAP recursively samples $x_{a_t} \sim p_\theta(x_{a_t}|y, \mathbf{x}_{<a_t})$ for $t = 1, \dots, T$ and uses each sample as the input to the next step in $\mathcal{M}$, as is standard for sequence generation models [12].

5.3 kNN Baseline

As a strong baseline for the parsing task, we consider a nearest neighbour classifier. We store our large dataset of samples $\mathcal{D} = \{(\tau^{(m)}, y^{(m)})\}_{m=1}^{N}$. At test time, given an input solution to parse, we can find its nearest neighbour in the samples with a linear search of $\mathcal{D}$, and return its associated trajectory. Depending on the problem, the solutions y will be in a different output space (image, text) and thus the distance metric used for the nearest-neighbour search will be domain dependent. We refer to this baseline as GG-kNN. Note that GG-kNN is quite costly in memory and runtime as it needs to store and iterate through all samples in the dataset.

6. EXPERIMENTS

We test generative grading on a suite of education data sets focusing on introductory courses from online platforms and large universities. For each dataset, we compare our approach to supervised learning, PCFGs, k-nearest neighbours, and human performance. In Sec. 6.1, we introduce the data sets, then present results in Sec. 6.3.

6.1 Datasets

We consider four educational contexts. Fig. 5 shows example student solutions for each problem.

Block-based Programming Code.org released a data set of student responses to eight Blocky exercises from one of their curriculums online, which focuses on drawing shapes with nested loops. We take the last problem in the curriculum (the most difficult one): drawing polygons with an increasing number of sides—which has 302 human graded responses with 26 misconceptions regarding looping and geometry (e.g. "missing for loop" or "incorrect angle") from [32].

Free Response Language Powergrading [1] contains 700 responses to a United States citizenship exam, each graded for

[4]Specific details can be found in the code.

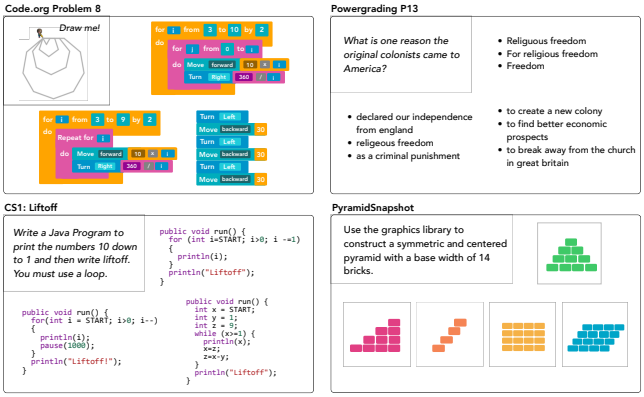


Figure 5: We show the prompt and example solutions for our four datasets.

correctness by 3 humans. Responses are in natural language, but are typically short (average of 4.2 words). We focus on the most difficult question, as measured by [24]: "name one reason the original colonists came to America". Correct responses span economic, political, and religious reasons.

Graphics Programming PyramidSnapshot is a university CS1 course assignment intended to be a student's first exposure to variables, objects, and loops. The task is to build a pyramid using Java's ACM graphics library by placing individual blocks. The dataset is composed of *images* of rendered pyramids from intermediary "snapshots" of student work. [33] annotated 12k unique snapshots with 5 categories representing "knowledge stages" of understanding.

University Programming Assignment Liftoff is a second assignment from a university CS1 course that tests looping. Students are tasked to write a program that prints a countdown from 10 to 1 followed by the phrase "Liftoff". In Sec. 7, we will use Liftoff for a human-in-the-loop study where experts generatively grade 176 solutions from a semester of students and measure accuracy and grading time.

6.2 Simulator Descriptions

We provide a brief overview of the Idea2Text simulators constructed for each domain.

Block-based Programming The primary innovation is to use the first decision node random variable to represents student ability. This ability variable will affect the distributions for random variables later in the trajectory such as deciding the loop structure and body. The intuition this captures is that high ability students make very few to no mistakes whereas low ability students tend to make many correlated misunderstandings. This simulator contains 52 decision nodes.

Free Response Language Idea2Text simulators over natural language need to explain variance in both semantic meaning and prose. We inspected the first 100 responses to gauge student thinking. Procedurally, the first random variable is choosing whether the production will be correct or incorrect. It then chooses a subject, verb, and noun dependent on the correctness. Correct answers lead to topics like religion, politics, and economics while incorrect answers are about taxation, exploration, or physical goods. Finally, we add a random variable to decide a writing style to craft a sentence. To capture variations in tense, we use a conjugator [7] for the final production. This simulator contains 53 decision nodes.

Graphics Programming The primary decision in this simulator decides between 13 "strategies" (e.g. making a parallelogram, right triangle, a brick wall, etc.) that the instructor believed students would use. Each of the 13 options leads to its own set of nodes that are responsible for deciding shape, location, and colour. The production uses Java to render an image output. This simulator contains 121 decision nodes, and required looking at 200 unlabelled student solutions in its design.

University Programming Assignment To model student thinking on Liftoff, this simulator first determines whether to use a loop, and, if so, chooses between "for" and "while" loop structures. It then formulates the loop syntax, choosing a condition statement and whether to count up or count down. Finally, it chooses the syntax of the print statements. Notably, each choice is dependent on previous ones. For example, choosing an end value in a for loop is sensibly conditioned on a chosen start value. This simulator contains 26 decision nodes.

6.3 Results for Feedback Prediction

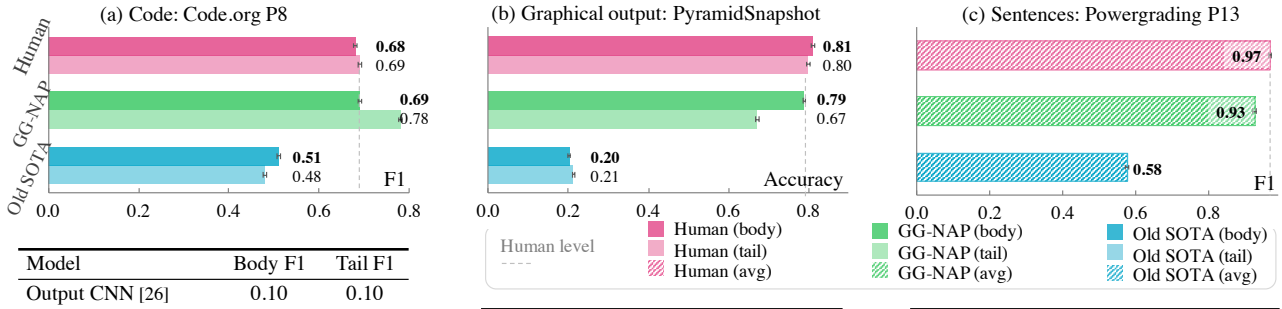
We show the results of generative grading for each of the datasets above.

For each dataset, we have access to a set of real student solutions and corresponding human-provided feedback labels, which we use for evaluation. We measure human accuracy relative to the majority label.

We ask instructors to create an Idea2Text simulator for each dataset, and train the deep inference network GG-NAP using simulated student solutions. At test time, we pass a real student solution into the inference model, and get back a trajectory of the simulator. This trajectory contains decision node choices that correspond to the human-provided feedback labels, and we use these as the predicted feedback.

Our performance metric for evaluating the model's predicted feedback labels is accuracy or F1 score, depending on the convention of prior work. Computing an average of the metric across the evaluation dataset would over-prioritise examples that appear frequently; this is particularly important to avoid for the Zipf distributed solutions. Since we care about providing feedback to struggling students in the tail of the distribution, we separately calculate performance for different "regions" of the Zipf. Specifically, we define the *head* as the k most popular solutions, the *tail* as solutions that appear only once or twice, and the *body* as the rest. As solutions in the head can be trivially memorised, we focus on performance on the body and tail.

Training Details We report averages over three runs; error bars are shown in Fig. 6. We use a batch size of 64, train for 20 epochs on 100k unique samples adaptively sampled from the simulator. We optimise using Adam [15] with a learning rate of 5e-4 and weight decay of 1e-7. For PyramidSnap-



Model	Body F1	Tail F1
Output CNN [26]	0.10	0.10
Program RNN [23]	0.27	0.22
MVAE [24]	0.38	0.26
Rubric Sampling [26]	0.51	0.48
GG-kNN	0.31	0.33
GG-NAP	**0.69**	**0.78**
Human	0.68	0.69

Model	Body Acc	Tail Acc
kNN [28]	0.20	0.12
NeuralNet [28]	0.20	0.21
GG-kNN	timeout	timeout
GG-NAP	**0.79**	**0.67**
Human	0.81	0.80

Model	Avg F1	Tail F1
Handcrafted [6]	0.58	-
T&N Best [17]	0.55	-
GG-kNN	0.78	0.63
GG-NAP	**0.93**	**0.76**
Human	0.97	0.90

Figure 6: Summary of results for providing feedback to student work in three educational contexts: block-based programming, graphics programming, and free response language. Generative grading shows strong performance in all three settings, closely approximating human-level performance in two data sets, and surpassing human-level performance in the other.

shot, we use VGG-11 [26] with Xavier initialisation [9] as the encoder network. For other data sets, we use a Recurrent Neural Network (RNN) with 4 layers, a hidden size of 256. The deep inference network itself is an unrolled RNN: we use a gated recurrent unit with a hidden dimension of 256 and no dropout. The value and index embedding layers output a vector of dimension 32. These hyperparameters were chosen using grid search.

Code.org As feedback for Code.org exercises has been studied in prior work [32], we compare generative grading to a suite of baselines including supervised models trained to classify misconceptions from the hand-labelled dataset (Output CNN [28] + Program RNN [32]), unsupervised models that learn a latent vector representation of student work (MVAE), to the k-nearest neighbours baseline GG-kNN from Sec. 5. Most relevant to our approach is the "rubric sampling" [32] comparison, which uses a PCFG to simulate students and generate a supervised data set to train a RNN classifier. Human accuracy is measured by comparing the feedback of multiple annotators to the majority label.

As shown in Fig. 6, generative grading is able to provide accurate feedback (historically measured as F1) beyond the level of individual human annotators, setting the new state-of-the-art. We observe a large improvement over prior work, which perform significantly worse than human graders. Compared to rubric sampling, we find a 18% (absolute) improvement in the body and a 30% (absolute) improvement in the tail. This clearly demonstrates the practical importance of being context-sensitive. The global state of Idea2Text simulators allow us to easily write richer generative models that are capable of better simulating real students. The potential impact of a human-level autonomous grader is large: Code.org is used by 610 million students, and our approach could save thousands of human hours for teachers by providing the same quality of feedback at scale.

Powergrading We find similarly strong performance on the Powergrading corpus of short answer responses to a citizenship question. Fig. 6 shows that generative grading reaches a F1 score of 0.93, an increase of 0.35 points above prior work that used hand-crafted features to predict correctness [6], and 0.38 points above supervised neural networks [24]. We were unable to compare to rubric sampling [31] as it was too difficult to write a faithful PCFG to describe free response language. Generative grading takes a large step towards closing the gap to human performance (F1 = 0.97). We are especially optimistic about these results as Powergrading responses contain natural language, this is promising signal that ideas from generative grading could generalise beyond computer science education.

PyramidSnapshot Investigating a third modality of image output from a graphics assignment, we find similar results comparing generative grading to the k-nearest neighbour baseline and a VGG image classifier presented in [33], outperforming the latter by nearly 50% absolute.

Unlike other datasets, the PyramidSnapshot dataset includes student's intermediary work, showing stages of progression through multiple attempts at solving the problem. With our near-human level performance, instructors could use GG-NAP to measure student cognitive understanding over time *as* students work. This builds in a real-time feedback loop between the student and teacher that enables a quick and accurate way of assessing teaching quality and characterising both individual and classroom learning progress. From a technical perspective, since PyramidSnapshot only includes rendered images (and not student code), generative grading was responsible for parsing student solutions from just images alone, a feat not possible without the flexibility of probabilistic programs used in Idea2Text. For this reason, we could not apply rubric sampling in this context either.

7. EXTENSIONS

Our results show that generative grading is a powerful tool for the feedback prediction task. However, our system is much more general than this and has many interesting extensions that we discuss in this section.

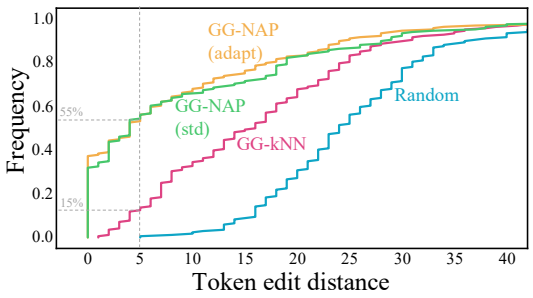


Figure 7: CDF of Levenshtein edit distance between student programs and nearest-neighbours using various algorithms.

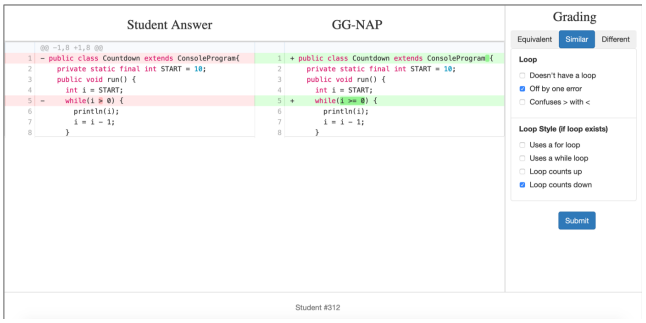


Figure 8: Human-in-the-loop Generative Grading UI

7.1 Nearest In-Simulator Neighbour

As described in Section 5, our inference model learns to map a given solution, y, to a decision trajectory in the simulator. So far, we have used this only to provide feedback with a fixed set of labels. However, the decision trajectory has a much more powerful interpretation; it represents the sequence of decisions in the simulator that the inference model thinks will produce y. Since we have the simulator in hand, we can actually execute it with these predicted sequence of decisions and inspect the simulated output, $\widehat{y}$.

If the simulated output is exactly equal to the original input solution, i.e. if $y = \widehat{y}$, then we can make a strong claim: the predicted trajectory from the inference model was provably correct and the corresponding labels can be assigned with 100% confidence. This is a claim that is seldom possible with traditional supervised learning methods and advances efforts towards creating explainable AI.

What about when $y \neq \widehat{y}$? In this case, the simulated output is not an exact match to the student solution, but we can still treat it as a "*nearest in-simulator neighbour*" to y. Fig. 7 shows the quality of these nearest neighbours to the student solutions using a distance metric like edit distance.

As we show below, these nearest neighbours can be used for powerful forms of feedback mechanisms.

7.2 Human-in-the-loop Grading

In a real-world setting, predicting feedback labels could be unreliable due to the high risk of giving students incorrect feedback. Beyond automated feedback, we explore how generative grading can be used to make human graders more effective using a human-in-the-loop approach.

To do this, we created a human-in-the-loop grading system using GG-NAP. For each student solution, we use the inference model to find the nearest in-simulator neighbour (Sec. 7.1); this nearest neighbour already has associated labels that are correct *for the nearest neighbour*. A human grader is presented with the original student solution, as well as a diff to the nearest neighbour; the grader then adjusts the labels of the nearest neighbour based on the diff to determine grades for the real solution. We show an image of the user-interface of this system in Fig. 8.

We investigated the impact of this human-in-the-loop system on grading accuracy and speed in a real classroom setting. We hired a cohort of expert graders (teaching assistants for a large private university) who graded 30 real student solutions to Liftoff. For control, half the graders proceeded traditionally, assigning a set of feedback labels by just inspecting the student solutions. The other half of graders additionally had access to (1) the feedback assigned to the nearest neighbour by GG-NAP and (2) a code differential between the student program and the nearest neighbour. Some example feedback labels included "off by one increment", "uses while loop", or "confused > with <". All grading was done on a web application that kept track of the time taken to grade a problem.

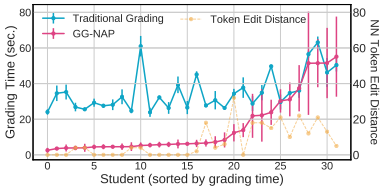
We found that the average time for graders using our system was *507 seconds* while the average time using traditional grading was *1130 seconds*, a more than double increase. Moreover, with our system, only *3 grading errors* (out of 30) were made with respect to gold-standard feedback given by the course professor, compared to the *8 errors* made with traditional grading. Fig. 9a shows these results for each of the 30 solutions.

The improved performance stems from the semantically meaningful nearest neighbours provided by GG-NAP. Having access to graded nearest neighbours helps increase grader efficiency and reliability by allowing them to focus on only "grading the diff" between the real solution and the nearest neighbour. By halving both the number of errors and the amount of time, GG-NAP can have a large impact in classrooms today, saving instructors and teaching assistants unnecessary hours and worry over grading assignments.

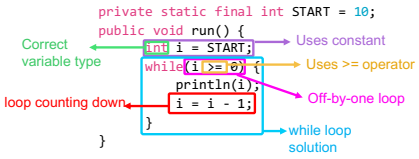
7.3 Highlighting feedback in student solutions.

The inferred decision trajectory for a student solution can also be used to provide "dense" feedback that highlights the section of the code or text responsible for each misunderstanding. This would be much more effective for student learning than vague error messages currently found on most online education platforms.

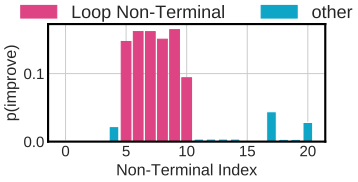
To achieve this, we leverage the fact that each decision node in the simulator gets recursively expanded to produce the final solution. This means it is easy to track the portions of the output that each decision node is responsible for. For decision nodes related to student confusions, we can highlight the portion of the output in the student solution which corresponds to this confusion. Fig. 9b shows a random pro-



(a) Classroom Experiment Results



(b) Automated Dense Feedback



(c) Auto-improving Simulators

Figure 9: (a) Plot of average time taken to grade 30 student solutions to Liftoff. Generative grading reduces grading time for 26 out of 30 solutions. The amount of time saved correlates with the token edit distance (yellow) to the nearest neighbour in the simulator. (b) Our approach allows for automatically highlighting which part of the student solution is responsible for a predicted misconception. (c) Given a Liftoff simulator that is missing a key "decrement loop" decision, we can automatically find decision nodes where inference often fails on real student solutions. The highest scoring decision nodes are all correctly related looping.

gram with automated, segment-specific feedback given by GG-NAP. This level of explainability is sorely needed in both education and AI.

7.4 Automatically Improving Simulators

Building Idea2Text simulators is an iterative process; a user wishing to improve their simulator would want a sense of where it is lacking. Fortunately, given a set of difficult examples where GG-NAP does poorly, we can deduce the decision nodes in the simulator that consistently lead to mistakes and use these to suggest components to improve.

To do this, for each nearest neighbour to a student solution we can find decision nodes that cause substring mismatches in the student solution, using regular expressions. This is possible because each decision node is responsible for a scoped substring in the nearest neighbour output solution (Sec. 7.3). By finding the decision nodes where the substring often differs between the neighbour and the solution, we can identify decisions that often causes mismatches.

To illustrate this, we took the Liftoff simulator, which contains a crucial decision node that decides between incrementing up or down in a "for" loop, and removed the option of incrementing down. We trained GG-NAP on this smaller simulator, and used a scoring mechanism to identify relevant decision nodes responsible for failing to parse student solutions that "increment down". Fig. 9c shows the distribution over which nodes GG-NAP believes to be responsible for the failed parses. The top 6 decisions that GG-NAP picked out all rightfully relate to looping and increments.

8. LIMITATIONS AND FUTURE WORK

Cost of writing good simulators. One of the most critical steps in our approach is the ability to write good Idea2Text simulators. Writing a good simulator does not require special expertise and can be undertaken by a novice in a short time. For instance, the PyramidSnapshot simulator that sets the new state of the art was written by a first-year undergraduate within a day. Furthermore, many aspects of simulators are re-usable: similar problems will share nonterminals and some invariances (e.g. the nonterminals that capture different ways of writing `for` loops are the same everywhere). This means every additional grammar is easier to write since it likely shares a lot in structure with existing grammars. Moreover, compared to weeks spent hand-labelling data, the cost of writing a grammar is orders of magnitude cheaper and leads to much better performance.

That being said, we believe there is room for interesting future work that explores how to make grammars easy to write and improve, with the extension in Sec. 7.4 already making some headway in this direction. There is also room for better formalising which types of problem domains can be faithfully modelled with Idea2Text simulators, and which domains are infeasible, like general essay writing. Lastly, more sophisticated inference approaches could be explored for handling semantic invariances in student output such as code reordering or variable renaming.

Connections to IRT. We find an interesting parallel of our work to Item Response Theory (IRT). IRT is essentially an extremely simple generative model that relates a student parameter θ to the probability of getting a question correct or incorrect. Some of our Idea2Text simulators also incorporate a student ability parameter θ to dictate likelihoods of making mistakes at different decisions, and can thus be seen as a more expressive and nuanced extension of the IRT generative model. Exploring this further is an interesting direction of research.

Generating questions with Idea2Text. We use Idea2Text simulators to model student decision-making and corresponding example solutions. This could be used to automatically generate example solutions with known issues to show students for pedagogical purposes. The Idea2Text library can also been used to generating questions corresponding to confusions instead of solutions corresponding to confusions.

9. CONCLUSION

We proposed a method for providing automated student feedback that showed promising results across multiple modalities and domains. Our proposed feedback system is capable of predicting student decisions corresponding to a given solution, allowing us to do nuanced forms of automated feedback. With it, "generative grading" can be used to automate feedback, visualise student approaches for instructors, and make grading easier, faster, and more consistent. Although more work needs to be done on making powerful grammars easier to write, we believe this is an exciting direction for the future of education and a step towards combining machine learning and human-centred artificial intelligence.

References

[1] S. Basu, C. Jacobs, and L. Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics*, 1:391–402, 2013.

[2] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, 20(28):1–6, 2019.

[3] W. G. Bowen. The 'cost disease'in higher education: is technology the answer? *The Tanner Lectures Stanford University*, 2012.

[4] J. S. Brown and K. VanLehn. Repair theory: A generative theory of bugs in procedural skills. *Cognitive science*, 4(4):379–426, 1980.

[5] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus. An adaptive sampling algorithm for solving markov decision processes. *Operations Research*, 53(1):126–139, 2005.

[6] J. Daxenberger, O. Ferschke, I. Gurevych, and T. Zesch. Dkpro tc: A java-based framework for supervised learning experiments on textual data. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66, 2014.

[7] S. Diao. mlconjug: A python library to conjugate verbs using machine learning techniques. *GitHub: https://github.com/SekouD/mlconjug*, 2018.

[8] M. Q. Feldman, J. Y. Cho, M. Ong, S. Gulwani, Z. Popović, and E. Andersen. Automatic diagnosis of students' misconceptions in k-8 mathematics. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[10] N. Goodman, V. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: a language for generative models. *arXiv preprint arXiv:1206.3255*, 2012.

[11] N. D. Goodman and A. Stuhlmüller. The design and implementation of probabilistic programming languages, 2014.

[12] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[13] S. Gulwani and R. Singh. Automated feedback generation for introductory programming assignments. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2013)*, pages 15–26, July 2013.

[14] Q. Hu and H. Rangwala. Reliable deep grade prediction with uncertainty estimation. *arXiv preprint arXiv:1902.10213*, 2019.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] K. R. Koedinger, N. Matsuda, C. J. MacLellan, and E. A. McLaughlin. Methods for evaluating simulated learners: Examples from simstudent. In *AIED Workshops*, 2015.

[17] T. A. Le, A. G. Baydin, and F. Wood. Inference compilation and universal probabilistic programming. *arXiv preprint arXiv:1610.09900*, 2016.

[18] J. Liu, Y. Xu, and L. Zhao. Automated essay scoring based on two-stage learning. *arXiv preprint arXiv:1901.07744*, 2019.

[19] X. Liu, S. Wang, P. Wang, and D. Wu. Automatic grading of programming assignments: An approach based on formal semantics. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering Education and Training*, ICSE-SEET '19, page 126–137. IEEE Press, 2019.

[20] T. G. MARTIN, M. A. BURGMAN, F. FIDLER, P. M. KUHNERT, S. LOW-CHOY, M. MCBRIDE, and K. MENGERSEN. Eliciting expert knowledge in conservation science. *Conservation Biology*, 26(1):29–38, 2012.

[21] H. Nilforoshan and E. Wu. Leveraging quality prediction models for automatic writing feedback. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12, 2018.

[22] A. O'Hagan. Eliciting expert beliefs in substantial practical applications. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(1):21–35, 1998.

[23] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Advances in neural information processing systems*, pages 505–513, 2015.

[24] B. Riordan, A. Horbach, A. Cahill, T. Zesch, and C. M. Lee. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–168, 2017.

[25] D. Ritchie, P. Horsfall, and N. D. Goodman. Deep Amortized Inference for Probabilistic Programs. Technical report, 2016.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] F. Wang and D. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical review letters*, 86:2050–3, 04 2001.

[28] L. Wang, A. Sy, L. Liu, and C. Piech. Learning to represent student knowledge on programming exercises using deep learning. In *EDM*, 2017.

[29] D. Wingate, A. Stuhlmueller, and N. Goodman. Lightweight implementations of probabilistic programming languages via transformational compilation. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 770–778, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings.

[30] B. Woods, D. Adamson, S. Miel, and E. Mayfield. Formative essay feedback using predictive scoring models. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 2071–2080, New York, NY, USA, 2017. Association for Computing Machinery.

[31] M. Wu, M. C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[32] M. Wu, M. Mosse, N. Goodman, and C. Piech. Zero shot learning for code education: Rubric sampling with deep learning inference. *arXiv preprint arXiv:1809.01357*, 2018.

[33] L. Yan, N. McKeown, and C. Piech. The pyramidsnapshot challenge: Understanding student process from visual output of programs. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 119–125, New York, NY, USA, 2019. ACM.

Knowledge Transfer by Discriminative Pre-training for Academic Performance Prediction

Byungsoo Kim, Hangyeol Yu, Dongmin Shin, Youngduck Choi
Riiid! AI Research
{byungsoo.kim,hangyeol.yu,dm.shin,youngduck.choi}@riiid.co

ABSTRACT

The needs for precisely estimating a student's academic performance have been emphasized with an increasing amount of attention paid to Intelligent Tutoring System (ITS). However, since labels for academic performance, such as test scores, are collected from outside of ITS, obtaining the labels is costly, leading to label-scarcity problem which brings challenge in taking machine learning approaches for academic performance prediction. To this end, inspired by the recent advancement of pre-training method in natural language processing community, we propose DPA, a transfer learning framework with **D**iscriminative **P**re-training tasks for **A**cademic performance prediction. DPA pre-trains two models, a generator and a discriminator, and fine-tunes the discriminator on academic performance prediction. In DPA's pre-training phase, a sequence of interactions where some tokens are masked is provided to the generator which is trained to reconstruct the original sequence. Then, the discriminator takes an interaction sequence where the masked tokens are replaced by the generator's outputs, and is trained to predict the originalities of all tokens in the sequence. We conduct extensive experimental studies on a real-world dataset obtained from a multi-platform ITS application and show that DPA outperforms the previous state-of-the-art generative pre-training method with a reduction of 4.05% in mean absolute error and more robust to increased label-scarcity.[1]

Keywords

Academic Performance Prediction, Deep Learning, Transfer Learning, Discriminative Pre-training

1. INTRODUCTION

Predicting a student's future academic performance is a fundamental task for developing modern Intelligent Tutoring System (ITS) which aims to provide personalized learning experience by supporting educational needs of each individual. However, labels for academic performance, such as test scores, are often scarce since they are external to ITS. For example, as shown in Figure 1, test scores are not automatically collected inside of ITS. Obtaining a test score requires a student to take the test in the designated test center, receive the score, and report the score to ITS. Transfer learning is a commonly taken approach to address such label-scarcity problems across different domains of machine learning. In this framework, a model is first pre-trained to optimize auxiliary objectives with abundant data, and then fine-tuned on the task of interest. In Artificial Intelligence in Education (AIEd) community, [3] introduced Assessment Modeling (AM), a set of pre-training tasks for label-scarce educational problems including academic performance prediction. AM proposed a pre-training method where first, a masked interaction sequence is generated by replacing a set of interactive features which can serve as criteria for pedagogical evaluation with artificial mask tokens. Then, given the

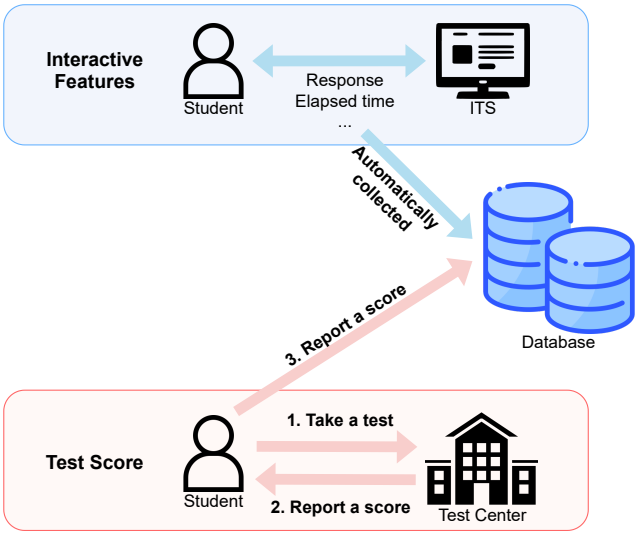


Figure 1: Interactive features, such as student response and elapsed time for the response, are automatically recorded to the database whenever a student interacts with ITS. On the other hand, more complicated steps are necessary to obtain a test score: a student should take the test in the designated test center, receive the test score, and report the score to ITS.

[1]For more detailed descriptions of experimental settings and results, please refer the arXiv version of this paper.

Byungsoo Kim, Hangyeol Yu, Dongmin Shin and Youngduck Choi "Knowledge Transfer by Discriminative Pre-training for Academic Performance Prediction". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 287-294. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

masked interaction sequence, a model is pre-trained to predict the masked interactive features. The idea was borrowed from the Masked Language Modeling (MLM) pre-training method proposed in [7]. In the MLM pre-training method, given a masked word sequence where some words in the sequence are replaced with an artificial mask token, a model is pre-trained to predict the masked words. However, recently, [6] pointed out that the MLM pre-training method has poor sample efficiency and suffers from pre-train/fine-tune discrepancy due to the artificial mask token, and proposed a new discriminative pre-training method. Considering the problems are also inherent in AM, potential gains are expected to be obtainable when the discriminative pre-training method is applied to academic performance prediction.

To this end, we propose DPA, a transfer learning framework with **D**iscriminative **P**re-training tasks for **A**cademic performance prediction. There are two models in DPA: a generator and a discriminator. In DPA's pre-training phase, the generator is trained to predict the masked interactive features in the same way as AM. Then, given a replaced interaction sequence which is generated by replacing the masked features with the generator's outputs, the discriminator is trained to predict whether each token in the sequence is the same as the one in the original interaction sequence. After the pre-training, the generator is thrown away and only the discriminator is fine-tuned on academic performance prediction. Also, we investigate diverse pre-training tasks for the generator and show that pre-training the generator to predict a student's response is more effective than to predict the correctness and timeliness of their response which were considered as the most pedagogical interactive features in AM. Extensive experimental studies conducted on a real-world dataset collected from a multi-platform ITS application show that DPA outperforms AM with a reduction of 4.05% in Mean Absolute Error (MAE) and more robust when the degree of label-scarcity increases.

2. SANTA: A SELF-STUDY SOLUTION EQUIPPED WITH AN AI TUTOR FOR ENGLISH EDUCATION

In this paper, we conduct experiments on a real-world dataset obtained from *Santa*[2], a multi-platform ITS with more than a million users in South Korea available through Android, iOS, and Web that exclusively focuses on the Test of English for International Communication (TOEIC) standardized examination. The publicly accessible version of the dataset was released under the name *EdNet* [4]. The TOEIC consists of two timed sections, Listening Comprehension (LC) and Reading Comprehension (RC). There are a total of 100 multiple choice exercises in each section, and the total score for each section is 495 in steps of 5 points. *Santa* provides learning experiences of solving exercises, studying explanations, and watching lectures. When a student consumes a specific learning content, *Santa* diagnoses their current academic status based on their learning activities records and recommends another learning content appropriate for their current position. *Santa* records diverse types of interactive features, such as student response, the duration of time the student took to respond, and the time interval between the current and previous learning activities. However, unlike the interactive features automatically collected from *Santa*, obtaining the official TOEIC score requires more steps: a student should register and pay for the test, take the test in the designated test center, receive the test score from the Educational Testing Service, and report the score to *Santa* (Figure 1). *Santa* collected students' TOEIC score data by offering small gifts to students when they report their scores.

[2]`https://aitutorsanta.com`

3. TRANSFER LEARNING FOR ACADEMIC TEST PERFORMANCE PREDICTION

To overcome the label-scarcity problem in academic test performance prediction, we consider burgeoning machine learning discipline of transfer learning. There is an open issue of what information to transfer or which pre-training task is the most effective for academic test performance prediction. Previous studies proposed two types of pre-training methods for AIEd Tasks: interaction-based method which models students' dynamic learning behaviors [13, 15, 8, 3], and content-based method which learns representations of learning contents [14, 23, 19, 24, 29]. [3] showed that interaction-based pre-training method outperforms content-based pre-training methods when the pre-trained model is fine-tuned on several label-scarce educational tasks including academic test performance prediction. Following this line of research, we propose a transfer learning framework where a model is pre-trained using only student interaction data, and fine-tune the pre-trained model on academic test performance prediction. In this paper, we consider the following interactive features:

- *eid*: A unique ID assigned to an exercise solved by a student. There are a total of 14419 exercises in the dataset.
- *part*: Each exercise belongs to a specific part that represents the type of the exercise. There are a total of 7 parts in the TOEIC.
- *response*: Since the TOEIC consists of multiple choice exercises and there are four options for each exercise, a student response for a given exercise is one of the options, 'a', 'b', 'c', or 'd'.
- *correctness*: Whether a student responded correctly to a given exercise. Note that *correctness* is a coarse version of *response* since *correctness* is processed by comparing *response* with a correct answer for a given exercise.
- *elapsed_time*: The amount of time a student spent on solving a given exercise.
- *timeliness*: Whether a student responded to a given exercise under the time limit. Note that *timeliness* is a coarse version of *elapsed_time* since *timeliness* is processed by comparing *elapsed_time* with the time limit recommended by domain experts for a given exercise.
- *exp_time*: The amount of time a student spent on studying an explanation for an exercise they had solved.
- *inactive_time*: The time interval between the current and previous interactions.

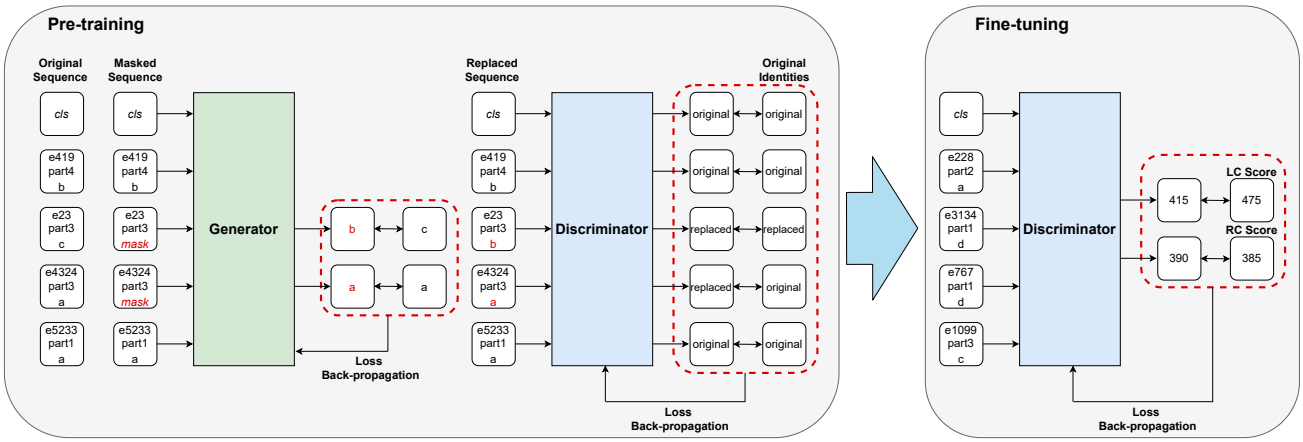


Figure 2: The overall pre-training/fine-tuning process of DPA when each token in an interaction sequence is a set of *eid*, *part*, and *response*, and *response* is a feature being masked. *mask* and *cls* are special tokens for mask and classification, respectively, which are the same as the ones used in [7].

In our experiments, we normalize the values of *elapsed_time*, *exp_time*, and *inactive_time* so they are between 0 and 1 to stabilize the training process.

4. PROPOSED METHOD

Figure 2 depicts our proposed method. There are two models in DPA: a generator and a discriminator. In pre-training phase, given a sequence of interactions $I = [I_1, \ldots, I_T]$, where each interaction $I_t = \{f_t^1, \ldots, f_t^k\}$ is a set of interactive features f_t^i, such as *eid*, *part*, and *response*, a masked interaction sequence $I^M = [I_1^M, \ldots, I_T^M]$ is generated by first randomly selecting a set of positions to mask $M = \{M_1, \ldots, M_m\}$ $(m < T)$, and for the masked position M_i, masking out a fixed set of features $\{f_{M_i}^1, \ldots, f_{M_i}^n\}$ $(n < k)$. For instance, in Figure 2, if the original interaction sequence is [(e419, part4, b), (e23, part3, c), (e4324, part3, a), (e5233, part1, a)] where each token in the sequence is a set of *eid*, *part*, and *response*, a masked interaction sequence where $M = \{2, 3\}$ and *response* as a masked feature is [(e419, part4, b), (e23, part3, *mask*), (e4324, part3, *mask*), (e5233, part1, a)]. Then, the generator takes the masked interaction sequence I^M as an input, and outputs predicted values O_{ij}^G for the masked features $f_{M_i}^j$. After that, a replaced interaction sequence $I^R = [I_1^R, \ldots, I_T^R]$ is generated by replacing the masked features $f_{M_i}^j$ with the generator's predictions O_{ij}^G. In Figure 2, since the generator's outputs for the masked features are 'b' and 'a', a replaced interaction sequence is [(e419, part4, b), (e23, part3, b), (e4324, part3, a), (e5233, part1, a)]. Then, the discriminator takes the replaced interaction sequence I^R as an input, and predicts whether each token in the sequence is the same as the one in the original interaction sequence (original) or not (replaced). After the pre-training, we throw away the generator and fine-tune the pre-trained discriminator on academic test performance prediction. We provide detailed explanations of each component in the generator and the discriminator, and training objective functions in the following subsections.

4.1 Interaction Embeddings

The embedding layer produces a sequence of interaction embedding vectors by mapping each interactive feature to an appropriate embedding vector. We take two different approaches to embed the interactive features depending on whether they are categorical (*eid*, *part*, *response*, *correctness*, and *timeliness*) or continuous (*elapsed_time*, *exp_time*, and *inactive_time*) variables. If an interactive feature is a categorical variable, we assign unique latent vectors to possible values of the feature including special values for mask (*mask*) and classification (*cls*). Take *response* as an example, there is an embedding matrix $E_{response} \in \mathbb{R}^{6 \times d_{emb}}$ where each row vector is assigned to one of 'a', 'b', 'c', 'd', *mask*, and *cls*. If an interactive feature is a continuous variable, we assign a single latent vector to the feature. Then, an embedding vector for the feature is computed by multiplying the latent vector and a value of the feature. For instance, we compute an embedding vector for *elapsed_time* as $et * E_{elapsed\_time}$, where et is a specific value and $E_{elapsed\_time} \in \mathbb{R}^{d_{emb}}$ is a latent vector assigned to *elapsed_time*. Also, mask and classification for the continuous interactive features are indicated by setting their values to -1 and 0, respectively. Not only embeddings for interactive features, positional embeddings are also incorporated into Transformer-based models [27] to consider chronological order of each token. Rather than using conventional positional embeddings which stores an embedding vector for every possible position, we adopt axial positional embeddings [17] to further reduce memory usage. The final interaction embedding vector of dimension d_{emb} for each time-step is the sum of all embedding vectors in the time-step. The interaction embedding layer is shared by both the generator and the discriminator.

4.2 Performer Encoder

Since its successful debut in Natural Language Processing (NLP) community, Transformer's attention mechanism has become a common recipe adopted across different domains of machine learning including speech processing [18], computer vision [1, 9], and AIEd [21, 2, 11, 22]. Compared to Recurrent Neural Network (RNN) family models, Transformer's attention mechanism has benefits of capturing longer-range

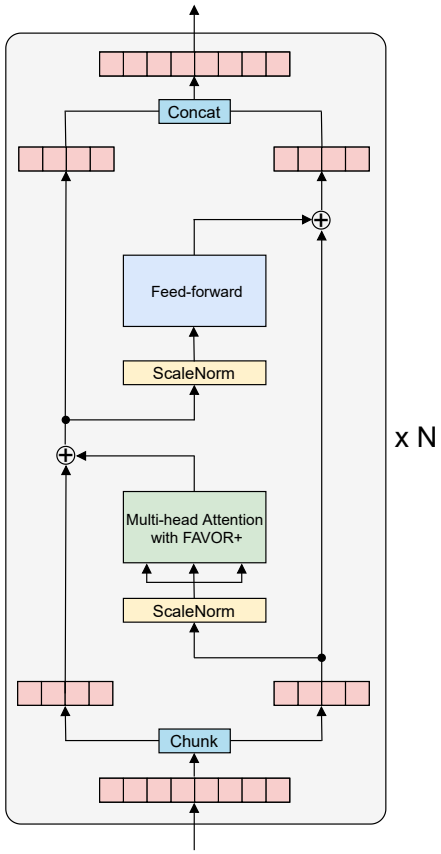


Figure 3: The reversible layer in the Performer encoder is composed of the FAVOR+-based multi-head attention layer and the point-wise feed-forward layer.

dependencies and allowing parallel training, which enables the model to achieve better performance with less training time. However, despite these advantages, the time and memory complexities of computing the attention grow quadratically with respect to input sequence length, requiring demanding computing resources for training the model on long sequences. For instance, if L is input sequence length and d is dimension of query, key, and value vectors, Transformer's attention is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d}}\Big)V,$$

where $Q, K, V \in \mathbb{R}^{L\times d}$. The time and memory complexities for computing $QK^\top$ in the above equation are $O(L^2 d)$ and $O(L^2)$, respectively. Therefore, the cost for training Transformer becomes prohibitive with large L, preventing training the model even on a single GPU.

The problem of improving the efficiency of Transformer's attention mechanism is a common concern of machine learning community. Recent studies have proposed several methods to reduce the computing complexities lower than the quadratic degree with respect to input sequence length [17, 28, 16, 25, 5]. In this paper, we adopt Performer [5] since it uses reasonable memory and makes a better trade-off between speed and performance [26]. Performer approximates attention kernels through Fast Attention Via positive Orthogonal Random features (FAVOR+) approach. For those who want to know more about FAVOR+, please refer [5].

With the efficient attention mechanism by FAVOR+, we propose the Performer encoder which is stacks of several identical reversible layers described in Figure 3. The reversible layer is based on Reversible Transformer [12, 17] architecture to further improve memory efficiency in backpropagation. An input of the reversible layer $x \in \mathbb{R}^{L\times d_{hidden}}$ is first chunked to $x_1, x_2 \in \mathbb{R}^{L\times d_{hidden}/2}$. Then, scaled l_2 normalization (ScaleNorm) [20] and FAVOR+-based multi-head attention layer (MultiHeadAttn) are applied to x_2, and the result is added to x_1 to compute $y_1 \in \mathbb{R}^{L\times d_{hidden}/2}$.

$$y_1 = x_1 + \text{MultiHeadAttn}(\text{ScaleNorm}(x_2)).$$

After that, the scaled l_2 normalization and point-wise feed-forward layer (FeedForward) are applied to y_1, and the result is added to x_2, computing $y_2 \in \mathbb{R}^{L\times d_{hidden}/2}$.

$$y_2 = x_2 + \text{FeedForward}(\text{ScaleNorm}(y_1)).$$

An output of the reversible layer $y \in \mathbb{R}^{L\times d_{hidden}}$ is a concatenation of y_1 and y_2. We stack the reversible layer multiple times to allow the final model to sufficiently represent underlying data distribution.

4.3 Generator

The generator computes hidden representations $[h_1^G, \ldots, h_T^G]$ by feeding the masked interaction sequence I^M to a series of the interaction embedding layer (InterEmbedding), a point-wise feed-forward layer (GenFeedForward1), the Performer encoder (GenPerformerEncoder), and another point-wise feed-forward layer (GenFeedForward2):

$$[I_1^{ME}, \ldots, I_T^{ME}] = \text{InterEmbedding}([I_1^M, \ldots, I_T^M])$$
$$[h_1^{GF}, \ldots, h_T^{GF}] = \text{GenFeedForward1}([I_1^{ME}, \ldots, I_T^{ME}])$$
$$[h_1^{GP}, \ldots, h_T^{GP}] = \text{GenPerformerEncoder}([h_1^{GF}, \ldots, h_T^{GF}])$$
$$[h_1^G, \ldots, h_T^G] = \text{GenFeedForward2}([h_1^{GP}, \ldots, h_T^{GP}]),$$

where $I_t^{ME}, h_t^G \in \mathbb{R}^{d_{emb}}$ and $h_t^{GF}, h_t^{GP} \in \mathbb{R}^{d_{gen\_hidden}}$. Then, depending on whether the masked features are categorical or continuous variables, generator outputs are computed differently. If the masked features are categorical variables, the outputs are sampled from a probability distribution defined by the following softmax layer:

$$O_{ij}^G \sim P_G(f_{M_i}^j | I^M) = \text{softmax}(E_j h_{M_i}^G).$$

If the masked features are continuous variables, the outputs are computed by the following sigmoid layer:

$$O_{ij}^G = \text{sigmoid}(E_j^\top h_{M_i}^G).$$

Similar to the case of categorical masked features, one can sample the outputs from a probability distribution defined by I^M and parameters of the generator when the masked features are continuous variables. For instance, the outputs can be sampled from the Gaussian distribution where the mean and the variance are determined by I^M and the generator's parameters. However, we make the outputs deterministic because sampling the outputs underperforms in our preliminary experiments when the masked features are continuous variables.

4.4 Discriminator

In pre-training, outputs of the discriminator $O^D = [O_1^D, \ldots, O_T^D]$ is computed by applying a series of the interaction embedding layer (InterEmbedding), a point-wise feed-forward layer (DisFeedForward1), the Performer encoder (DisPerformerEncoder), and another point-wise feed-forward layer (DisFeedForward2) to the replaced interaction sequence I^R:

$$[I_1^{RE}, \ldots, I_T^{RE}] = \text{InterEmbedding}([I_1^R, \ldots, I_T^R])$$
$$[h_1^{DF}, \ldots, h_T^{DF}] = \text{DisFeedForward1}([I_1^{RE}, \ldots, I_T^{RE}])$$
$$[h_1^{DP}, \ldots, h_T^{DP}] = \text{DisPerformerEncoder}([h_1^{DF}, \ldots, h_T^{DF}])$$
$$[O_1^D, \ldots, O_T^D] = \text{DisFeedForward2}([h_1^{DP}, \ldots, h_T^{DP}]),$$

where $I_t^{RE} \in \mathbb{R}^{d_{emb}}$, $h_t^{DF}, h_t^{DP} \in \mathbb{R}^{d_{dis\_hidden}}$, $O_t^D \in \mathbb{R}$, and the sigmoid is applied to the last layer of the discriminator. After the pre-training, we slightly modify the discriminator by replacing the last layer with a layer having appropriate dimension for academic test performance prediction.

4.5 Training Objectives

The objective for pre-training is to minimize the following loss function:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} \text{GenLoss}(O_{ij}^G, f_{M_i}^j) + \lambda \sum_{t=1}^{T} \text{DisLoss}(O_t^D, \mathbb{1}(I_t^R = I_t)),$$

where GenLoss is the cross entropy (or mean squared error) loss function if the masked features are categorical (or continuous) variables, DisLoss is the binary cross entropy loss function, and $\mathbb{1}$ is the identity function. For ease of notation, we omit an index for each input sample in the above equation. If there are more than one masked features in each time-step ($n > 1$), the generator is trained under the multi-task leaning scheme. The objective for fine-tuning is to minimize the mean squared error loss between the model's predictions and score labels.

5. EXPERIMENTS

5.1 Effects of Generator's Pre-training Tasks

There are multiple interactive features to be masked in each token of the interaction sequence, which raises a question of how to construct a set of masked interactive features, and accordingly, which pre-training task for the generator is the most effective for academic test performance prediction. By default, all interactive features listed in Section 3 are taken as inputs for both the generator and discriminator. However, if *response* (or *elapsed_time*) is masked, *correctness* (or *timeliness*) is excluded from the inputs and vice versa since there is an overlap of information that the features represent. For example, when both *response* and *correctness* are taken as inputs, and *correctness* is masked, the generator can predict the masked *correctness* by only looking at *eid* and *response* without considering other interactions, which leads to poor pre-training. The results are described in Table 1.

The best result was obtained under the pre-training task of predicting *response* alone, which is slightly better than that of predicting *correctness*, and both *response* and *correctness*. Predicting correctness of student response is an important task in AIEd as can be seen from the large volume of studies about Knowledge Tracing. Also, [3] empirically showed that student response correctness is the most pedagogical interactive feature for academic test performance prediction. However, rather than pre-training a model to predict whether a student correctly responded to a given exercise, the pre-training task of predicting student response itself injects more fine-grained information into the model, which leads to the more effective pre-training for academic test performance prediction. Interestingly, the underperformed results were obtained when predicting *elapsed_time* or *timeliness* in pre-training despite the benefits their information bring to several AIEd tasks [10, 30, 22]. We hypothesize that *elapsed_time* and *timeliness* may introduce irrelevant noises and thus guide the model towards a direction inappropriate for academic test performance prediction. In the case of *exp_time* and *inactive_time*, we observed that the generator failed to learn to predict their values when only given the interactive features listed in Section 3, which leads to unstable pre-training. From these observations, in the following subsections, we conduct experimental studies based on the pre-training task of predicting *response* alone.

Table 1: Comparison between different pre-training tasks.

Pre-training task	MAE
response	**50.65** $\pm$ 1.26
response + *elapsed_time*	54.86 $\pm$ 1.64
response + *timeliness*	52.91 $\pm$ 1.38
response + *exp_time*	57.54 $\pm$ 1.47
response + *inactive_time*	60.69 $\pm$ 1.74
correctness	51.36 $\pm$ 0.97
correctness + *elapsed_time*	53.36 $\pm$ 1.43
correctness + *timeliness*	52.60 $\pm$ 1.20
correctness + *exp_time*	54.36 $\pm$ 1.62
correctness + *inactive_time*	55.04 $\pm$ 1.58
response + *correctness*	51.13 $\pm$ 1.60
response + *correctness* + *elapsed_time*	52.15 $\pm$ 1.43
response + *correctness* + *timeliness*	53.05 $\pm$ 1.81
response + *correctness*+ *exp_time*	53.09 $\pm$ 1.25
response + *correctness* + *inactive_time*	56.41 $\pm$ 1.72

5.2 DPA vs. Baseline Methods

We compare DPA with the following pre-training methods:

- No pre-training: We train the fine-tuning models only on the fine-tuning dataset.
- Autoencoding: Autoencoding (AE) is a generative pre-training method widely used across different domains of machine learning including AIEd [13, 8]. Given an unmasked interaction sequence, AE pre-trains a model to reconstruct the input interaction sequence.
- Assessment Modeling: Assessment Modeling (AM) [3] is the previous state-of-the-art generative pre-training method for academic test performance prediction. In AM, a model takes a masked interaction sequence as an input and is pre-trained to predict masked features. AM is exactly the same as fine-tuning the pre-trained generator in DPA.

Also, we investigate whether DPA is effective with the following different fine-tuning models:

Table 2: Comparison of DPA with baseline methods.

Pre-training method	Fine-tuning model	MAE
No pre-training	MLP	82.89 ± 3.23
	BiLSTM	84.05 ± 2.06
	Transformer encoder	107.06 ± 2.52
	Performer encoder	81.76 ± 1.24
AE	MLP	79.46 ± 1.15
	BiLSTM	85.64 ± 1.89
	Transformer encoder	75.13 ± 3.10
	Performer encoder	64.80 ± 1.43
AM	MLP	77.17 ± 2.14
	BiLSTM	58.16 ± 1.28
	Transformer encoder	57.16 ± 2.08
	Performer encoder	52.79 ± 1.39
DPA	MLP	77.24 ± 1.59
	BiLSTM	57.59 ± 1.76
	Transformer encoder	55.99 ± 1.62
	Performer encoder	$\mathbf{50.65} \pm 1.26$

- MLP: Multi-Layer Perceptron (MLP) is stacks of simple fully-connected layers. Given an interaction sequence, interaction embedding vectors of all time-steps are summed together to compute a fixed-dimensional vector which is fed to a series of the fully-connected layers.
- BiLSTM: Bi-directional Long Short-Term Memory (BiLSTM) is a model widely used for time series data prediction tasks. The global max pooling layer is applied on top of the BiLSTM layer to obtain a fixed-dimensional intermediate representation from an input sequence of varying length.
- Transformer Encoder: Transformer Encoder is a series of several identical layers composed of a multi-head self-attention layer with the softmax attention kernel and a point-wise feed-forward layer. We set the Transformer encoder's attention window size to 512 due to the out of GPU memory occuring when training the Transformer encoder of 1024 attention window size on our single GPU machine.

As described in Table 2, transferring the pre-trained knowledge brings better results in most cases, and the best result is obtained from DPA. Especially, when the Performer encoder, the best performing fine-tuning model, is used as the fine-tuning model, DPA reduces MAE by 4.05%, 21.84%, and 38.05% compared to AM, AE, and No pre-training, respectively. Among the baseline pre-training methods excluding No pre-training, the worst result is obtained from AE beacuse the pre-training task of AE is much easier than that of AM and DPA. We observed that the loss curve of AE converged to near zero within the first pre-training evaluation.

5.3 Robustness to Increased Label-scarcity

Since the motivation behind our proposal of DPA is the label-scarcity problem, we investigate how MAE changes at varying degrees of label-scarcity. Figure 4 and Table 3

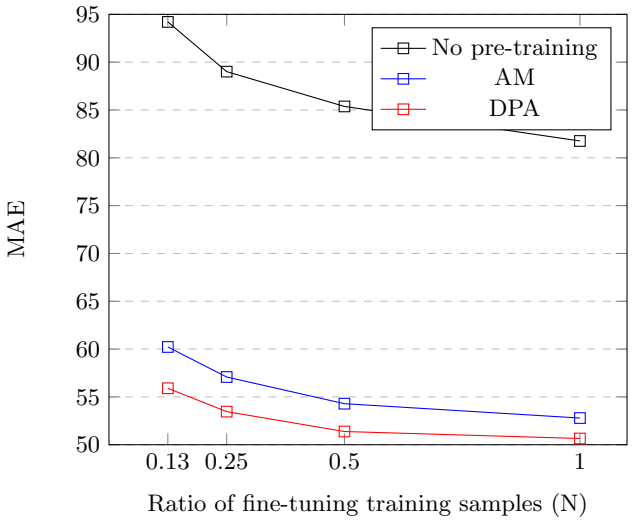


Figure 4: The black, blue, and red lines represent MAEs for No pre-training, AM, and DPA, respectively, when the number of fine-tuning training samples becomes 1/2, 1/4, and 1/8 of the entire dataset.

Table 3: Comparison of DPA with AM and No pre-training at varying degrees of label-scarcity.

N	No pre-training	AM	DPA
1/8	94.21 ± 8.40	60.22 ± 1.86	55.90 ± 1.97
1/4	89.01 ± 2.14	57.08 ± 1.75	53.46 ± 1.45
1/2	85.37 ± 1.15	54.29 ± 1.50	51.38 ± 1.16
Full	81.76 ± 1.24	52.79 ± 1.39	50.65 ± 1.26

describe the results when using 1/2, 1/4, and 1/8 of the total number of fine-tuning training samples. In all degrees of label-scarcity, DPA consistently outperforms AM. Also, DPA fine-tuned on 1/2, 1/4, and 1/8 of the dataset outperforms AM fine-tuned on the entire dataset, 1/2, and 1/4 of the dataset, respectively, which shows that DPA is more robust to label-scarcity than AM. Compared with No pre-training, the gap between No pre-training and the other two pre-training methods increases as the number of labels becomes scarce. Furthermore, the other two pre-training methods fine-tuned on 1/8 of the dataset outperform No pre-training fine-tuned on the entire dataset.

6. CONCLUSION

In this paper, we proposed DPA, a transfer learning framework with discriminative pre-training tasks for academic performance prediction. Our experimental results showed the effectiveness of DPA for the label-scarce academic performance prediction task over the previous state-of-the-art generative pre-training method. Avenues of future research include investigating more effective pre-training tasks for academic performance prediction and pre-train/fine-tune relations in AIEd.

7. REFERENCES

[1] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. Generative pretraining

from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

[2] Y. Choi, Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and J. Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pages 341–344, 2020.

[3] Y. Choi, Y. Lee, J. Cho, J. Baek, D. Shin, S. Lee, Y. Cha, B. Kim, and J. Heo. Assessment modeling: Fundamental pre-training tasks for interactive educational systems. *arXiv preprint arXiv:2002.05505*, 2020.

[4] Y. Choi, Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, C. Bae, B. Kim, and J. Heo. Ednet: A large-scale hierarchical dataset in education. In *International Conference on Artificial Intelligence in Education*, pages 69–73. Springer, 2020.

[5] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[6] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] M. Ding, Y. Wang, E. Hemberg, and U.-M. O'Reilly. Transfer learning using representation learning in massive open online courses. In *Proceedings of the 9th international conference on learning analytics & knowledge*, pages 145–154, 2019.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] M. Feng, N. Heffernan, and K. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User modeling and user-adapted interaction*, 19(3):243–266, 2009.

[11] A. Ghosh, N. Heffernan, and A. S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2330–2339, 2020.

[12] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. *arXiv preprint arXiv:1707.04585*, 2017.

[13] B. Guo, R. Zhang, G. Xu, C. Shi, and L. Yang. Predicting students performance in educational data mining. In *2015 International Symposium on Educational Technology (ISET)*, pages 125–128. IEEE, 2015.

[14] Z. Huang, Q. Liu, E. Chen, H. Zhao, M. Gao, S. Wei, Y. Su, and G. Hu. Question difficulty prediction for reading problems in standard tests. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[15] X. J. Hunt, I. K. Kabul, and J. Silva. Transfer learning for education data. In *Proceedings of the ACM SIGKDD Conference, El Halifax, NS, Canada*, volume 17, 2017.

[16] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[17] N. Kitaev, Ł. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[18] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6706–6713, 2019.

[19] Q. Liu, Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, and G. Hu. Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, 33(1):100–115, 2019.

[20] T. Q. Nguyen and J. Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

[21] S. Pandey and G. Karypis. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*, 2019.

[22] D. Shin, Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi. Saint+: Integrating temporal features for ednet correctness prediction. *arXiv preprint arXiv:2010.12042*, 2020.

[23] Y. Su, Q. Liu, Q. Liu, Z. Huang, Y. Yin, E. Chen, C. Ding, S. Wei, and G. Hu. Exercise-enhanced sequential modeling for student performance prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[24] C. Sung, T. I. Dhamecha, and N. Mukhi. Improving short answer grading using transformer-based pre-training. In *International Conference on Artificial Intelligence in Education*, pages 469–481. Springer, 2019.

[25] Y. Tay, D. Bahri, L. Yang, D. Metzler, and D.-C. Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR, 2020.

[26] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[28] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[29] Y. Yin, Q. Liu, Z. Huang, E. Chen, W. Tong, S. Wang, and Y. Su. Quesnet: A unified representation for heterogeneous test questions. In *Proceedings of the 25th ACM SIGKDD International*

Conference on Knowledge Discovery & Data Mining, pages 1328–1336, 2019.

[30] L. Zhang, X. Xiong, S. Zhao, A. Botelho, and N. T. Heffernan. Incorporating rich features into deep knowledge tracing. In *Proceedings of the fourth (2017) ACM conference on learning@ scale*, pages 169–172, 2017.

Toward Improving Student Model Estimates through Assistance Scores in Principle and in Practice

Napol Rachatasumrit
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
napol@cmu.edu

Kenneth R. Koedinger
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
koedinger@cmu.edu

ABSTRACT

Student modeling is useful in educational research and technology development due to a capability to estimate latent student attributes. Widely used approaches, such as the Additive Factors Model (AFM), have shown satisfactory results, but they can only handle binary outcomes, which may yield potential information loss. In this work, we propose a new partial credit modeling approach, PC-AFM, to support multi-valued outcomes. We focus particularly on the amount of assistance, that is, the number of error feedback and hint messages, a student needs to get a problem step correct. Because errors and hint requests may not only derive from student ability, but also from non-cognitive factors (e.g., students may game the system), we first test PC-AFM on synthetic data where this source of variation is not present. We confirm that PC-AFM is indeed better than AFM in recovering the true student and knowledge component (KC) parameters and even predicts student error rates better than a model fit to error rates. We then apply the approach to six real-world datasets and find that PC-AFM outperforms AFM in reliable estimation of KC parameters and produces better generalization to new students, which requires better KC estimates. However, consistent with the hypothesis that student assistance behavior is driven by motivational or meta-cognitive factors beyond their ability, we found that PC-AFM was not better in reliable estimation of student parameters nor in generalization across items, which requires accurate student estimates. We propose *cross-measure cross-validation* as a general method for comparing alternative measurement models for the same desired latent outcome.

Keywords

Additive Factors Model, Student Modeling, Model Comparison, Learning Curves

Napol Rachatasumrit and Kenneth Koedinger "Toward Improving Student Model Estimates through Assistance Scores in Principle and in Practice". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 295-301. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

1. INTRODUCTION

Student modeling has been an important tool that researchers can use to estimate latent student abilities. Similarly, intelligent tutoring systems also depend on how accurately we can predict student mastery to deliver efficient adaptive learning. Current popular approaches, such as Additive Factors Model (AFM) [4, 18, 13] and Bayesian Knowledge Tracing (BKT) [5, 13], perform reasonably well by including the growth factors in their models. However, they are restricted by using only binary student performance (e.g. correct/incorrect response), which could suffer from an information loss due to its dichotomized nature.

For example, many existing intelligent tutoring systems (ITS) support step-by-step interactions [22], which usually allow students to try multiple attempts or request for hints until they are able to complete the step correctly. These interactions are important for an ITS because it allows the system to provide immediate feedback or support an adaptive experience, while collecting a rich interaction dataset on student actions. However, since AFM and BKT can only handle binary outcomes, the student data is needed to be aggregated through a rollup procedure before we can use it in student modeling. This means only success on students' first attempt on each step will be included in the data, and the rest of the actions (e.g. other attempt or hint requests) will be ignored. To illustrate how this could be problematic, let's imagine student A who had one incorrect attempt on a step before correctly completing it and student B who had multiple incorrect attempts and asked for multiple hints on the same step before getting it right. The dichotomous model like AFM and BKT would treat both students as the same on this particular step, but we can see that it is more likely that student A has demonstrated better knowledge than student B.

In our case, we are concerned with having a raw measure of student success at each assessment opportunity. There are different functions for producing or deriving an outcome measure for the data available in a tutoring system. Perhaps the most typical function is: first transaction correct = 1; otherwise = 0 where both hints and incorrect responses are both counted as a failure. While there are multiple ways to elicit polytomous outcomes from ITS student data, in this work we focus on an assistant score, which is a total number of incorrect attempts and hint requests combined for each step. From our preliminary analysis, we found that

there are correlations between assistance scores and AFM's predicted error rate, which suggests that there could be an extra information in assistance scores compared to a binary correctness outcome.

In this work, we are interested in whether or not an assistance score model could be a better predictor of student's change in performance than a dichotomous model like AFM. Particularly, our research questions are: (1) How can we develop an effective statistical measurement model that uses assistance scores? and (2) How do we compare two different response models?

A popular approach to compare different cognitive models in Educational Data Mining is to use goodness-of-fit (e.g. Bayesian Information Criterion), but it is not applicable in our scenario because our model is based on different outcomes (correctness vs assistance score). Alternative versions of measures of predictor variables can be contrasted through cross validations, but it becomes inadequate when the outcome variables are different. We also discuss a set of strategies for addressing the general problem of how to compare alternative measurement models for the same desired latent outcome. Particularly, how do we compare a binary correctness model with a polytomous Assistance Score model?

We propose a new cognitive modeling approach to support polytomous outcomes and demonstrated its ability to recover parameters and predict student error rates better than AFM in synthetic data. We then evaluated our model to six real-world datasets spanning five different domains from the DataShop repository [10]. We found that our model outperforms AFM in most Student-blocked CVs and estimating KC parameters, but it falls short at estimating student intercepts. We hypothesize that our model is struggling to estimate student parameters in the real-world datasets due to variance in students' help-seeking behavior, such as gaming-the-system, that leads to the extra variance in Assistance Scores above and beyond the variance associated with student ability.

2. RELATED WORK

2.1 Item Response Theory with Partial Credit

Item Response Theory (IRT) models [6] is the preferred method used in several state assessments in the United States and international assessments [8]. The goal of the IRT model is to estimate the latent construct (e.g. student ability) and item characteristics (item difficulty) based on only a collection of responses.

The simplest variation of IRT is the Rasch model (1PL model) [19], which is characterized by a single parameter representing item difficulty (d_j), and a single parameter representing student ability (a_i). As Eq.1 is equivalent to a logistic function, the Rasch model is essentially a logistic regression model.

$$p(r_{ij} = 1) = \frac{1}{1+^{-(a_i - d_j)}} \quad (1)$$

Other variations increase the complexity by introducing extra parameters. For example, the 2PL model adds a discrimination parameter for each item that controls the slope of the logistic function, and the 3PL model that also includes a pseudo-guessing parameter for each item. Even though, these models are characterized by a different number of parameters, they are all based on dichotomous response data (e.g. correctness). There is another class of IRT models that can be applied to polytomous outcomes, where each response can be a different value [17, 21]. An example of responses that is applicable to this class of models are Likert scale. There are different variations of polytomous IRT models, such as Partial Credit Model (PCM) [14], Generalized Partial Credit Model (GPCM) [15], and Graded Response Model (GRM) [20].These polytomous models are generalized from the dichotomous IRT models and can be reduced to the dichotomous IRT models when there are only two response categories. Our model extends the polytomous model to include growth factor by applying a similar approach to PCM to AFM.

2.2 Knowledge Tracing Approaches

Intelligent tutoring systems (ITS) have been shown to be effective in improving student learning outcomes across different domains [2, 9], and mastery learning strategies have been an important component in these systems. To implement mastery learning, knowledge tracing techniques are regularly utilized by ITSs [7] to adaptively assess students' knowledge states, which is used to decide when students have mastered skills and are ready to move on to other skills.

In many existing ITSs, such as Cognitive Tutor Authoring Tools (CTAT) [1], students are given a number of practice opportunities for each skill , and students are usually allowed to try multiple attempts or request for hints until they are able to successfully complete the step on each practice opportunity. The goal of a knowledge tracing algorithm when used for mastery learning is to determine when to stop giving students practice opportunities for the given skill.

Knowledge tracing is often performed by a statistical model of student learning that could be fit to data. There are two popular families of methods [12]: Bayesian Knowledge Tracing (BKT) [5, 13] and Additive Factors Model (AFM) [4, 18, 13]. Both methods include growth factors in order to estimate students' performance as it is changing with learning. BKT models student knowledge as a latent variable in a Hidden Markov Model. AFM is an extension of the IRT model that includes learning opportunity counts in the model. Even though these methods have been proven to work well in many scenarios, they are based on the binary error measurement model (correct or incorrect) and thus do not make use of potential added information from the number of error and hint messages a student may receive. Our approach explores this opportunity by extending AFM to use such multi-valued or polytomous outcomes in hopes of better estimating student knowledge. While other variations on AFM, such as Performance Factor Analysis (PFA) [18] and individualized AFM (iAFM) [13], have been shown in some cases to produce better prediction fit than AFM, we chose to use AFM to simplify the contrast between binary and polytomous measurement models and with the goal of producing more parsimonious and interpretable parameter estimates. Future work can explore alternatives.

2.3 DataShop Data Features

In this work, we use a variety of real world datasets across different domains from the DataShop repository [10]. LearnLab's DataShop (http://learnlab.org/datashop) is an open data repository of educational data with associated visualization and analysis tools, which has data from thousands of students derived from interactions with on-line course materials and intelligent tutoring systems.

In DataShop terminology, Knowledge Components (KCs) are used to represent pieces of knowledge, concepts or skills that students need to solve problems [11]. When a specific set of KCs are mapped to a set of instructional tasks (usually steps in problems) they form a KC Model, which is a specific kind of student model.

Each dataset in DataShop consists of a set of student transactions, which is a collection of students' interactions with ITSs. The collected students' actions include (but not limited to) correct attempts, incorrect attempts, and hint requests. The transactions that belong to the same practice opportunity get aggregated into a single students' step through the rollup procedure. The correctness of the step depends on the result of the student's first response for the practice opportunity, and the total number of incorrect attempts and hint requests is reported as an Assistance Score of the step. Most existing knowledge tracing algorithms use students' steps, rather than transactions, in their models.

3. METHOD

The Additive Factors Model (AFM) [4] is a logistic regression that extends Item Response Theory by incorporating a growth or learning term. The model gives the probability p_{ij} that a student i will get a problem step j correct based on the student's baseline ability (θ_i), the baseline difficulty of the related KCs on the problem step (β_k), and the learning rate of the KCs (γ_k). The learning rate represents the improvement on a KC with each additional practice opportunity, so it is multiplied by the number of practice opportunities (T_{ik}) that the student already had on the KC.

$$log(\frac{p_{ij}}{1-p_{ij}}) = \theta_i + \Sigma_k(q_{jk}\beta_k + q_{jk}\gamma_k T_{ik}) \tag{2}$$

Our extension of AFM to support a polytomous outcome measure, like Assistance Score, is inspired by the Partial Credit Model (PCM) [14], which is an adjacent-categories logit model [21]. The model was designed to work with ordered polytomous response categories with a specific order or ranking of responses, which is the case for Assistance Score. It is widely applied in aptitude testing to allow for partial credit for near correctness of a response. In adjacent-categories logit models, we model the odds of a higher category relative to the adjacent lower one, and this paired comparison creates the ordering of the categories.

Assistance Score can be interpreted in the partial credit framework as follows. A student who gets a problem step correct on their first try or after fewer errors or hint requests is more likely to have the associated competence than a student who makes many errors or requests multiple hints before getting the step correct. Thus, students making no errors and needing no hints get full credit (Assistance Score = 0) and students with errors and/or hint requests get partial credit in rough proportion to the number hint and errors.

The Partial Credit Additive Factors Model (PC-AFM) builds upon these two different statistical models, AFM and PCM. For a student i and a step j, there is a set of probabilities $P_{ij} = \{p_{ija}; a = 0, 1, ..., A\}$ describing the chance for student i to get Assistance Score a on the step j, where A is the maximum Assistance Score. In this work, we decided to limit an Assistance Score at 5 because values above this tend not to be meaningful and rare, but extreme outliers (e.g., where assistance score is over 20 or even 140!) would significantly bias the model. 98% of our data have an Assistance Score of 5 or less. We extend AFM to use multivariate generalized linear mixed model, and the link function in logistic regression takes the vector-valued form.

$$f_{link}(P_{ij}) = \begin{pmatrix} f_{link,1}(P_{ij}) \\ ... \\ f_{link,A}(P_{ij}) \end{pmatrix} = \begin{pmatrix} log(\frac{p_{ij1}}{p_{ij0}}) \\ ... \\ log(\frac{p_{ijA}}{p_{ijA-1}}) \end{pmatrix} \tag{3}$$

Note that $f_{link,0}$ is not included due to the number of non-redundant probabilities. PC-AFM use adjacent-categories logits as a link function based on PCM. The ath adjacent-categories logit is the logit of getting an Assistance Score a versus $a-1$. Each link function is an extended version of AFM's linear model (Eq. 2) with a level parameter (α_a), which represents the difficulty to improve from an Assistance Score a to $a-1$.

$$f_{link,a}(P_{ij}) = \theta_i + \alpha_a + \Sigma_k(q_{jk}\beta_k + q_{jk}\gamma_k T_{ik}) \tag{4}$$

Inverting this function gives an expression for the probabilities of student i to complete a problem step j with each of the possible Assistance Scores a.

$$\begin{aligned} p_{ija} &= \frac{e^{\lambda_a}}{\Sigma_{i=0}^{A} e^{\lambda_i}} \\ \lambda_a &= \begin{cases} 0 & \text{if } a = 0 \\ \Sigma_{l=1}^{a} f_{link,l}(P_{ij}) & \text{otherwise} \end{cases} \end{aligned} \tag{5}$$

4. EXPERIMENT

We conduct experiments on both synthetic data and real student data to evaluate the performance of PC-AFM. We used the synthetic data to validate PC-AFM's parameter recovery capability and examine our evaluation strategy in a synthetic environment in which Assistance Score is stochastically derived from student ability alone. In particular, Assistance Scores in the synthetic data are not confounded by other student variations, such as their motivational state. We hypothesized that PC-AFM would work less effectively with the real student data because of non-ability effects on Assistance Score, such as students' help seeking strategies or propensity to game the system.

While goodness-of-fits metrics, such as BIC, are widely used

to compare different cognitive models [16], such as knowledge tracing algorithms, it is not applicable in our case due to the difference of outcome measures between AFM and PC-AFM. The challenge is how we can compare models that are based on different outcomes (error rate vs Assistance Score), while targeting the same desired latent measure (e.g. student's ability).

We explore two strategies to tackle this comparison problem. The first approach is to use parameter estimate reliability in split-half comparisons. Since both AFM and PC-AFM share the majority of their parameters (student intercepts, KC intercepts, and KC slopes), we can compare their parameter recovery capability. However, unlike synthetic data, the true parameters are not known in real data, so we need to use the reliability of parameter estimates in split-half comparisons instead. Another strategy is to compare cross-measure predictions. The assumption is that if a model based on polytomous outcomes (Assistance Score) yields better accuracy than a model based on binary outcomes (error rate) in predicting both polytomous and binary outcomes, the polytomous model will be demonstrated to be a better measurement model. This strategy is applicable in our scenario because there are connections between both outcomes. Since a student step is considered correct only when there is no assistance, the error rate can be derived by calculating the probability of Assistance Score = 0. On the other hand, we can convert the error rate to a probability of an Assistance Score by calculating the likelihood, where given an error rate p, the probability of having an Assistance Score a is $(1-p)p^a$. Then we can use CVs on both measures to compare the models.

4.1 Experiment 1: Synthetic Data

In order to validate PC-AFM capability to recover student and KC parameters, we synthetically generate datasets of student steps based on a logistic regression model. Given a set of student and KC parameters together with an opportunity count, a distribution over Assistance Scores is determined. We then sample once from the distribution to generate an Assistance Score of that student step. We generated 6 datasets of varying numbers of students and KCs, of which the true student and KC parameters are known, to examine parameter recovery capacity of PC-AFM in comparison to AFM. In each generated dataset, student intercepts range from -2 to 2, KC intercepts range from -1 to 1, and KC slopes range from 0 to 0.5. The number of KCs ranges from 8 to 32, and the number of students range from 25 to 200.

We also evaluate both models with three types of cross-validation (CV), Random (data points are split randomly), Student-blocked (data points are split by student), and Item-blocked (data points are split by item), to demonstrate if our model training on Assistance Score, can outperform a dichotomous model training on error rate in predicting dichotomous outcomes.

Table 1: Correlation between true and estimated parameters in synthetic data.

Dataset	Stu Intercept		KC Intercept		KC Slope	
	PC	AFM	PC	AFM	PC	AFM
KC8_S25	**0.978**	0.954	**0.996**	0.802	**0.914**	0.675
KC8_S50	**0.973**	0.936	**0.998**	0.985	**0.972**	0.964
KC8_S100	**0.973**	0.931	**1.000**	0.984	**0.952**	0.909
KC8_S200	**0.975**	0.936	**1.000**	0.979	**0.975**	0.735
KC16_S50	**0.990**	0.977	**0.998**	0.780	**0.962**	0.933
KC32_S50	**0.996**	0.988	**0.995**	0.799	**0.929**	0.543

Table 2: Correlation between split-halves parameters in synthetic data

Dataset	Stu Intercept		KC Intercept		KC Slope	
	PC	AFM	PC	AFM	PC	AFM
KC8_S25	**0.932**	0.828	**0.990**	0.895	**0.912**	0.498
KC8_S50	**0.963**	0.906	**0.998**	0.931	**0.972**	0.945
KC8_S100	**0.980**	0.941	**0.998**	0.850	**0.969**	0.888
KC8_S200	**0.871**	0.790	**0.999**	0.955	**0.910**	0.894
KC16_S50	**0.947**	0.857	**0.997**	0.947	**0.927**	0.843
KC32_S50	**0.967**	0.942	**1.000**	0.883	**0.997**	-0.345

We report on results for each of six different synthetic datasets by comparing PC-AFM and AFM. We found that PC-AFM better recovers the true student and KC parameters than AFM in almost all comparisons using correlation (Table 1). All contrasts are the same using mean absolute error. As the number of students goes up, both models tend to better recover the true parameters. The correlations of parameters in split-half comparison are reported in Table 2, which show a similar pattern to the correlation between estimated and true parameters. This demonstrates that the parameter correlation in split-half comparisons, which can be computed in real data, is a reasonable proxy for true parameter recovery, which cannot be computed in real data.

Figure 1 illustrates better true parameter recovery using Assistance Score and PC-AFM than using error rate and AFM. PC-AFM parameter estimates (red x's) are generally accurate across the spectrum of known parameter values (x-axis), as can be seen by their closeness to the line, which is identity function (intercept of 0, slope of 1). AFM estimates (blue dots) are generally biased toward the extremes. For student intercepts (Figure 1a), low prior knowledge students are estimated by error rate/AFM to be worse than they are and high prior knowledge students are estimated to be better than they are. For KC intercepts (Figure 1b), hard KCs (on the left) are estimated by error rate/AFM to be even harder than are. For hard KCs, most responses are errors, yielding quite low estimates by error rate/AFM. But, these same steps show more variance in Assistance Score/PC-AFM as somewhat better students and higher opportunities will produce lower, but non-zero Assistance Scores (i.e., not changing in error rate).

In error rate CV results, except Item-blocked CV where both models perform similarly, PC-AFM outperforms AFM in all other CVs (Table 4). Recall that these CV evaluations require PC-AFM, while fit to Assistance Score (polytomous outcome), to predict error rate (dichotomous outcome). When we turn the tables and compare methods on predicting Assistance Score, we find a similar pattern where PC-AFM yields better accuracy in most CVs (Table 3).

4.2 Experiment 2: Real student data

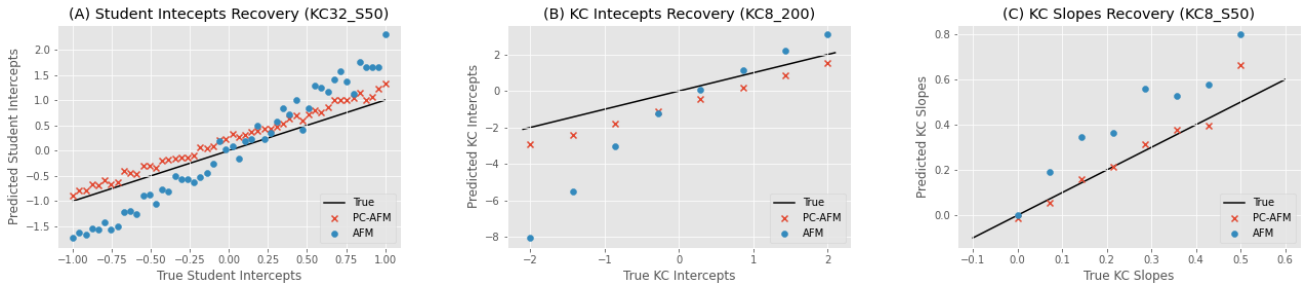


Figure 1: Using Assistance Score and PC-AFM on synthetic data produces better estimates of the true parameters, for all three of student intercepts, KC intercepts, and KC slopes than does using error rate and AFM.

Table 3: Cross-validation results (RSME) in synthetic data predicting Assistance Score in the test set by estimating parameters based on Assistance Score (PC-AFM) or on Error Rate (AFM) in the training set.

Dataset	Random		Stu-Blocked		Item-Blocked	
	PC	AFM	PC	AFM	PC	AFM
KC8_S25	**0.546**	0.598	**0.542**	0.600	**0.586**	0.634
KC8_S50	**0.544**	0.599	**0.541**	0.601	**0.575**	0.610
KC8_S100	**0.536**	0.596	**0.532**	0.599	**0.550**	0.602
KC8_S200	**0.541**	0.597	**0.537**	0.600	**0.541**	0.597
KC16_S50	**0.540**	0.600	**0.537**	0.601	**0.566**	0.604
KC32_S50	**0.540**	0.587	**0.539**	0.590	**0.579**	0.626

In the second experiment, we examine PC-AFM across a variety of real world datasets. We used 6 datasets across different domains (statistics, English articles, algebra, and geometry) from the DataShop repository. Table 5 shows the number of students, items, KCs, total transactions for each dataset. For each dataset, we use the KC model that achieves the best BIC reported on the DataShop repository. All KC models coded a single KC per step. The number of KCs ranges from 9 to 64, and the number of students ranges from 52 to 318.

For each dataset, we evaluated both PC-AFM and AFM on 5 independent runs of 3-fold CVs of each type predicting both Assistance Score and error rate. We report the result of Assistance Score CVs in Table 6 and the results of error rate CVs in Table 7. We found that PC-AFM outperforms AFM in Student-blocked in both Assistance Score and error

Table 4: Cross-validation results (RSME) in synthetic data predicting Error Rate in the test set by estimating parameters based on Assistance Score (PC-AFM) or on Error Rate (AFM) in the training set.

Dataset	Random		Stu-Blocked		Item-Blocked	
	PC	AFM	PC	AFM	PC	AFM
KC8_S25	**0.275**	0.278	0.310	**0.306**	**0.370**	0.430
KC8_S50	**0.273**	0.280	**0.282**	0.304	0.356	**0.297**
KC8_S100	**0.273**	0.277	**0.283**	0.300	**0.387**	0.449
KC8_S200	**0.271**	0.275	**0.278**	0.295	**0.278**	0.282
KC16_S50	**0.277**	0.281	**0.278**	0.311	0.301	**0.294**
KC32_S50	**0.287**	0.291	**0.292**	0.320	0.358	**0.347**

Table 5: Real Student Dataset.

Dataset	Domain	#Stu	#Item	#KC
ds308	College Statistics	52	113	9
ds313	English articles	120	85	26
ds372	English articles	99	84	15
ds388	Middle School math	318	64	64
ds392	Geometry	123	2035	43
ds394	English articles	97	180	13

rate CVs in most datasets, which suggests that PC-AFM can achieve better estimates of KC parameters. To validate the hypothesis, we investigated split-halves parameters correlation of both models. We splitted the datasets on students to evaluate KC slopes and intercepts correlation, and we splitted the datasets on KCs to evaluate students' intercepts (Table 8). On average, PC-AFM yields better correlations of both KC intercepts (0.954 vs 0.946) and KC slopes (0.600 vs 0.563), but correlations of student intercepts is significantly higher for AFM (0.784 vs 0.495).

5. DISCUSSION

Assistance score should, in principle, improve model parameter estimates and predictions based on them. A student who gets a step correct after just one error or one hint (Assistance Score = 1) is likely to be closer to full acquisition of a KC than a student who makes an error and requests 3 hints (Assistance Score = 4). However, the error rate metric commonly used with BKT and AFM treats these the same, since the student was not correct on their first attempt at the step without a hint. Thus, there is potentially extra in-

Table 6: Cross-validation results (RSME) in real data predicting Assistance Score in the test set by estimating parameters based on Assistance Score (PC-AFM) or on Error Rate (AFM) in the training set.

Dataset	Random		Stu-Blocked		Item-Blocked	
	PC	AFM	PC	AFM	PC	AFM
ds308	0.376	0.376	0.381	**0.378**	**0.384**	0.388
ds313	0.541	**0.528**	**0.551**	0.554	**0.549**	0.555
ds372	0.478	**0.463**	**0.480**	0.481	**0.484**	0.487
ds388	0.672	**0.649**	**0.682**	0.703	**0.702**	0.703
ds392	0.385	**0.354**	**0.386**	0.387	**0.385**	0.390
ds394	0.499	**0.486**	0.499	0.499	**0.504**	0.510

Table 7: Cross-validation results (RSME) in real data predicting Error Rate in the test set by estimating parameters based on Assistance Score (PC-AFM) or on Error Rate (AFM) in the training set.

Dataset	Random		Stu-Blocked		Item-Blocked	
	PC	AFM	PC	AFM	PC	AFM
ds308	0.336	**0.326**	0.332	**0.328**	0.341	**0.339**
ds313	0.417	**0.408**	**0.413**	0.440	0.435	**0.424**
ds372	0.379	**0.377**	**0.383**	0.402	0.388	**0.387**
ds388	0.454	**0.421**	0.439	**0.470**	0.501	**0.456**
ds392	0.324	0.324	**0.325**	0.333	0.325	0.325
ds394	0.395	**0.391**	**0.388**	0.418	0.403	0.403

formation about students' level of knowledge acquisition in the Assistance Score not present in error rate. On the other hand, prior research, for example on gaming the system [3], suggests there are other reasons students may produce repeated incorrect entries or hint requests. These may produce enough confounding variance to make using Assistance Score worse at accurate latent parameter estimation than using error rate.

In developing a statistical model, PC-AFM, to convert Assistance Scores to knowledge acquisition estimates, we first wanted to confirm that PC-AFM works as intended and is able to benefit from extra information in Assistance Score when no confounding sources for Assistance Score variation are present. Indeed, when we generate synthetic data where Assistance Scores are stochastically produced from known latent parameters, we demonstrate better parameter recovery using Assistance Score and PC-AFM than using error rate and AFM. As shown in Figure 1, PC-AFM estimates of student parameters are better correlated with true parameters and the AFM estimates are baised at the extremes.

This parameter recovery method for comparing these two different measurement models cannot be applied to real datasets because the true parameters are unknown. Thus, we employed we explored two other approaches: parameter estimate reliability and our novel cross-measure cross-validation approach. We demonstrated better parameter estimate reliability (in split-halves comparisons) using PC-AFM than AFM. We also show how it is possible to use cross-measure predictions to evaluate which of two different measurement models works better, call them M1 and M2. We show that estimating based on M1 (e.g., assistant score) can predict M2 (e.g., error rate) on held-out data better than estimating based on M2 itself (e.g., error rate). We believe this cross-measure cross-validation is a novel approach for comparing measurement models.

Assessing whether Assistance Score is a better measure than Error Rate in real student data is complicated in two ways. First, we do not have access to the true parameters in real datasets, so we turn to measures of reliability and predictive validity. Second, we know from models of gaming the system and help seeking that students may produce Assistance Scores for motivational and metacognitive reasons that are potentially independent of a mastery source. In other words, Assistance Scores have a student-driven source of variation that may reduce their effectiveness in estimating student mastery. We hypothesize that our model is struggling to estimate student parameters in the real-world datasets due to variance in students' help seeking behavior.

Table 8: Split-halves parameters correlation in real data.

Dataset	Stu Intercept		KC Intercept		KC Slope	
	PC	AFM	PC	AFM	PC	AFM
ds308	0.113	**0.486**	**0.971**	0.955	**0.745**	0.583
ds313	0.490	**0.830**	**0.948**	0.937	0.865	**0.905**
ds372	0.427	**0.803**	**0.985**	0.968	0.433	**0.639**
ds388	0.567	**0.873**	**0.946**	0.945	0.225	**0.354**
ds392	0.830	**0.901**	**0.973**	0.964	**0.494**	0.485
ds394	0.541	**0.809**	0.904	**0.906**	**0.838**	0.413

We found that in real world datasets PC-AFM can better estimate KC parameters than AFM, which results in PC-AFM outperforming AFM in Student-blocked CVs. KC parameters estimates significantly impact Student-blocked CVs because they are the sole driver of these predictions. Poor student estimates do not impact Student-blocked CVs because they are not carried from the training to test as blocking means there are different students in the test than training. It does impact Random CVs and Item-blocked CVs because they are likely to have some students showing up in both test and training.

6. CONCLUSION AND FUTURE WORK

We investigated whether or not Assistance Score provides a better measurement model than error rate for estimating student's ability. To pursue this question, we developed a statistical model, PC-AFM, that utilizes Assistance Score. We also faced the more general problem of how to compare alternative measurement models for the same desired latent outcome. In typical model comparison the predicted outcome measure stays the same, but such comparison does not work when the outcome measures are different. We proposed two strategies to tackle this problem: parameter estimate reliability in split-halves comparisons and a new approach we call, cross-measure cross-validation. We demonstrated that these strategies work well by using synthetic data to show that a model that better recovers parameters will also yield better results with these strategies.

We demonstrated that PC-AFM outperforms AFM when Assistance Scores are synthesized to be meaningful, but its performance is hindered by non-ability variance in students' behavior in the real-world datasets. Future work can explore this finding by synthesizing Assistance Scores that derive from both ability and motivational factors.

Future work can also test our measurement model comparison strategies. For example, while it has been standard practice in many tutoring systems to count hints as errors (M1), some have wondered whether it would be better to not count hints as errors (M2). Our measurement model comparison techniques, split-half reliability and cross-measure cross-validation, can be used to compare M1 and M2 to infer which provides better estimates of student ability.

7. REFERENCES

[1] V. Aleven, B. M. McLaren, J. Sewall, and K. R. Koedinger. The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. In *International Conference on Intelligent Tutoring Systems*, pages 61–70. Springer, 2006.

[2] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207, 1995.

[3] R. Baker, J. Walonoski, N. Heffernan, I. Roll, A. Corbett, and K. Koedinger. Why students engage in "gaming the system" behavior in interactive learning environments. *Journal of Interactive Learning Research*, 19(2):185–224, 2008.

[4] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International Conference on Intelligent Tutoring Systems*, pages 164–175. Springer, 2006.

[5] R. S. d Baker, A. T. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International conference on intelligent tutoring systems*, pages 406–415. Springer, 2008.

[6] S. E. Embretson and S. P. Reise. *Item response theory*. Psychology Press, 2013.

[7] Y. Gong and J. E. Beck. Towards detecting wheel-spinning: Future failure in mastery learning. In *Proceedings of the second (2015) ACM conference on learning@ scale*, pages 67–74, 2015.

[8] W. Harlen. The assessment of scientific literacy in the oecd/pisa project. 2001.

[9] K. R. Koedinger, J. R. Anderson, W. H. Hadley, M. A. Mark, et al. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8(1):30–43, 1997.

[10] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the edm community: The pslc datashop. *Handbook of educational data mining*, 43:43–56, 2010.

[11] K. R. Koedinger, A. T. Corbett, and C. Perfetti. The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science*, 36(5):757–798, 2012.

[12] K. R. Koedinger, S. D'Mello, E. A. McLaughlin, Z. A. Pardos, and C. P. Rose. Data mining and education. *Wiley Interdisciplinary Reviews: Cognitive Science*, 6(4):333–353, 2015.

[13] R. Liu and K. R. Koedinger. Towards reliable and valid measurement of individualized student parameters. *International Educational Data Mining Society*, 2017.

[14] G. N. Masters. A rasch model for partial credit scoring. *Psychometrika*, 47(2):149–174, 1982.

[15] E. Muraki. A generalized partial credit model: Application of an em algorithm. *ETS Research Report Series*, 1992(1):i–30, 1992.

[16] A. A. Neath and J. E. Cavanaugh. The bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(2):199–203, 2012.

[17] R. Ostini and M. L. Nering. *Polytomous item response theory models*. Number 144. Sage, 2006.

[18] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger. Performance factors analysis–a new alternative to knowledge tracing. *Online Submission*, 2009.

[19] G. Rasch. Studies in mathematical psychology: I. probabilistic models for some intelligence and attainment tests. 1960.

[20] F. Samejima. Graded response model. In *Handbook of modern item response theory*, pages 85–100. Springer, 1997.

[21] F. Tuerlinckx and W.-C. Wang. Models for polytomous data. In *Explanatory Item Response Models*, pages 75–109. Springer, 2004.

[22] K. VanLehn. The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265, 2006.

Learning Expert Models for Educationally Relevant Tasks using Reinforcement Learning

Christopher J. MacLellan
College of Computing and Informatics
Drexel University
Philadelphia, PA 19104
christopher.maclellan@drexel.edu

Adit Gupta
College of Computing and Informatics
Drexel University
Philadelphia, PA 19104
ag3338@drexel.edu

ABSTRACT

There has been great progress towards Reinforcement Learning (RL) approaches that can achieve expert performance across a wide range of domains. However, researchers have not yet applied these models to learn expert models for educationally relevant tasks, such as those taught within tutoring systems and educational games. In this paper we explore the use of Proximal Policy Optimization (PPO) [25] for learning expert models for tutoring system tasks. We explore two alternative state and action space representations for this RL approach in the context of two intelligent tutors (a fraction arithmetic tutor and a multicolumn addition tutor). We compare the performance of these models to a computational model of learning built using the Apprentice Learner architecture. To evaluate these models, we look at whether they achieve mastery and how many training opportunities they take to do so. Our analysis shows that at least one PPO model is able to successfully achieve mastery within both tutors, suggesting that RL models might be successfully applied to learn expert models for educationally relevant tasks. We find that the Apprentice model also achieves mastery, but requires substantially less training (thousands of times less examples) than PPO. Finally, we find that there is an interaction between the PPO representation and task (one representation is better for one tutor and the other representation is better for the other tutor), suggesting that the design of the state and action representations for RL is important for success. Our work showcases the promise of RL for expert model discovery in educationally relevant tasks and highlights limitations and challenges that need further research to overcome.

Keywords

Reinforcement Learning, Simulated Students, Expert Model Authoring

1. INTRODUCTION

Researchers have made great progress towards developing Reinforcement Learning (RL) models that can meet or exceed human skill at complex tasks across a broad range of domains. For example, the recently developed Proximal Policy Optimization (PPO) algorithm [25] can learn to play a broad range of Atari games at an expert level through trial and error. A team of five PPO-trained models can beat a team of five human professional champions at DOTA2, a collaborative online multiplayer battle arena game [4]. Researchers have applied a related RL approach called A3C [19] to develop agents that can beat top human experts at Starcraft2, a multiplayer real-time strategy game [28]. Finally, RL has been applied widely to the area of robotics and autonomous systems; e.g., RL models can fly an F16 to beat a expert human pilots in simulated 1v1 dogfights [7].

Despite these successes, there has been surprisingly little work exploring the applicability of RL to educationally relevant tasks, such as those found in K12 or higher education. We do not mean that RL has not been applied to support learning and education; in fact, there is a large amount of work exploring how RL can be applied to optimize students instructional sequences [9]. However, we assert that there has been very little exploration of how emerging RL approaches perform on the kinds of educationally relevant tasks that humans often engage in; e.g., learning math.

Given this gap we might ask, what are the benefits of applying RL to educationally relevant tasks? The recent work on computational models of learning highlights many possible benefits. First, machine learning agents can support researchers and instructional designers in authoring cognitive models [17, 30] and discovering knowledge component models [14] that can drive personalized learning technologies. Although RL models utilize different representations than more traditional expert-system models (e.g., statistical representations), learned models do still represent an expert model.[1] Thus, tutors might apply these models to provide feedback on student behaviors. Researchers and designers might also use machine learning agents to cognitively crash test instruction before more costly human trials [16, 15, 31].

Given these benefits, why has more work not explored the use of recent RL methods for these tasks? One possibility is that applying RL to educational tasks is not straightforward. There exist toolkits, like GymAI [5], MuJoCoEnv [23], and PyBullets [6], for interfacing RL algorithms with simulation

[1] An expert model maps states to correct next action(s).

Christopher Maclellan and Adit Gupta "Learning Expert Models for Educationally Relevant Tasks using Reinforcement Learning". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 302-309. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

environments and games platforms like Atari, StarCraft2, and DOTA2. These toolkits have powered the explosion of RL research. Unfortunately, no such interfaces exist for educational tasks, such as those found in intelligent tutoring systems or educational games. Also, many educational tasks do not fit cleanly into the standard RL paradigm of observing the state, choosing an action, receiving a reward, and repeating; e.g., many tutors let learners request hints and worked examples, which RL systems cannot leverage.[2]

Beyond these challenges, it is possible that the tasks themselves are less attractive for RL research. For example, educationally relevant tasks, such as fraction arithmetic, seem simple when compared to tasks such as flying an F16 or playing DOTA2. It is possible that these tasks do not challenge current RL systems. However, it is also possible that these tasks present challenges that have prevented researchers from successfully applying RL to them. For example, tutor tasks often have much larger action spaces than game-based tasks where RL has been successfully applied (e.g., actions for inputting the numbers 1-50 in any interface field vs. six buttons on an Atari controller).

In this paper, we set out to investigate these ideas and to lay a foundation for future research programs to apply RL to the kinds of educationally relevant tasks that humans regularly engage with. Our goal is not to show that RL provides a good model of human learning and behavior (we do not think that it does). Instead, we simply aim to show how RL methods might be applied to tasks relevant to human learning. Our hope is that RL approaches might offer new means for authoring and evaluating educational technologies.

To support these investigations we present TutorGym, an open-source toolkit for interfacing RL agents (as well as other kinds of machine learning agents) with intelligent tutoring system tasks. This toolkit lets us apply RL models to two educational environments: a fraction arithmetic tutor and a multicolumn addition tutor. We developed two PPO models that vary in their features and action representations. We also compared these models to a previously developed Apprentice Learner model [16], which is a more cognitively inspired model of how people learn from examples and feedback within intelligent tutors. We conducted a factorial study design where we applied these three models to our two educational tasks. Our key findings are:

1. The PPO models are able to achieve mastery at these tasks, suggesting that they do generalize from games and robotics tasks to educationally relevant tasks;
2. The PPO models require much more training than our Apprentice Learner model to achieve mastery (thousands of times more training), even when we provide PPO with the same background knowledge as Apprentice (the PPO-Operator variant). This suggests that human-like models, such as Apprentice, are more efficient than PPO.
3. We find that there is an interaction between PPO's representation and the task, suggesting that representation is central to RL performance and that it needs to be tailored for each task.

[2]Inverse RL [1] can learn from expert examples, but typically this is done offline in batch rather than interactively interleaved with RL.

We claim that there is an synergistic opportunity to do research at the intersection of RL and education that has not yet been fully explored and this paper aims to lay the foundation for these future explorations. There are many potential ways that educational data mining and learning analytic communities might benefit from the development and use of RL models, such as PPO. Similarly, there is an opportunity to improve RL by exploring its application to the kinds of educationally relevant learning tasks that humans engage in during K12 and higher education.

2. BACKGROUND

2.1 OpenAI Gym

OpenAI Gym is an open-source toolkit for RL development [5]. Gym provides an standardized interface for applying RL to tasks. An environment created with Gym has standardized state and action descriptions and supports methods, for querying the state, taking an action, and collecting rewards. Gym currently supports multiple environments such as robot simulations or Atari games. Our research builds on Gym, so that we can directly interface existing RL implementations with educationally relevant tasks without having to create custom implementations.

2.2 Proximal Policy Optimization (PPO)

PPO is a deep RL algorithm that was recently developed by OpenAI [25]. It is a policy gradient method that achieves state-of-the-art performance across many tasks. We chose PPO over alternatives, such as TRPO [24] and ACER [29], because it supports a broad range of state and action representations and is much easier to tune than alternatives. For this work, we use the stable-baselines3 implementation of the PPO algorithm, which has verified performance on multiple RL benchmarks [22].

2.3 Apprentice Learner Architecture

The open-source Apprentice Learner Architecture [16] generalizes prior simulated student models [13, 18] and provides a platform for investigating and comparing alternative simulated student models. Apprentice models have been successfully applied to learn expert models for 8 different tutor tasks spanning multiple domains (math, language, chemistry, and engineering) [17]. Emerging work explores the use of Apprentice models for supporting domain experts, such as teachers, in authoring tutors through teaching rather than programming [30]. In this work, we use one of the standard Apprentice models as a baseline for evaluating the PPO models because it have been successfully applied in previous work to learn expert models for educationally relevant tasks. For a complete description of this model see [17].

3. TUTORGYM

To support the development of machine learning agents we created TutorGym, a toolkit that provides a machine interfaces for multiple tutor environments.[3] TutorGym leverages the Gym [5] to enable existing RL implementations (that support Gym) to interface with these environments.

Our toolkit extends Gym to enable agents to request worked examples. Tutors generate both next-step hints and feedback, so the examples are automatically generated by the

[3]We have open-sourced TutorGym under an MIT license and made it publicly available here: https://tutorgym.ai

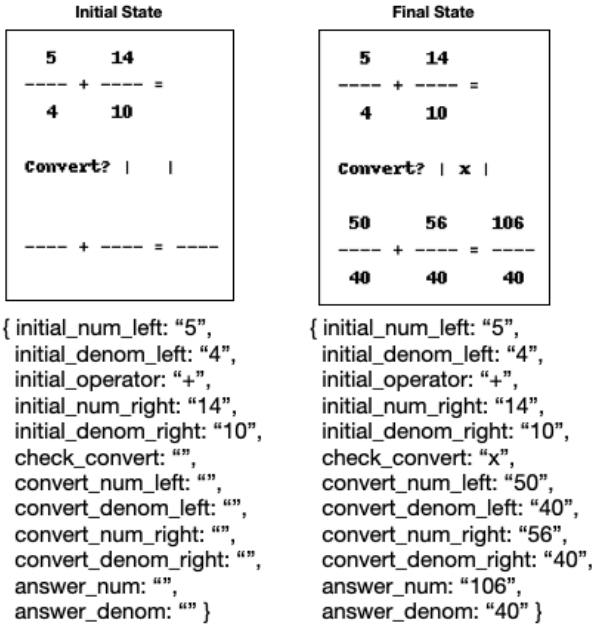


Figure 1: Fractions tutor, as rendered within TutorGym with its underlying base feature representation.

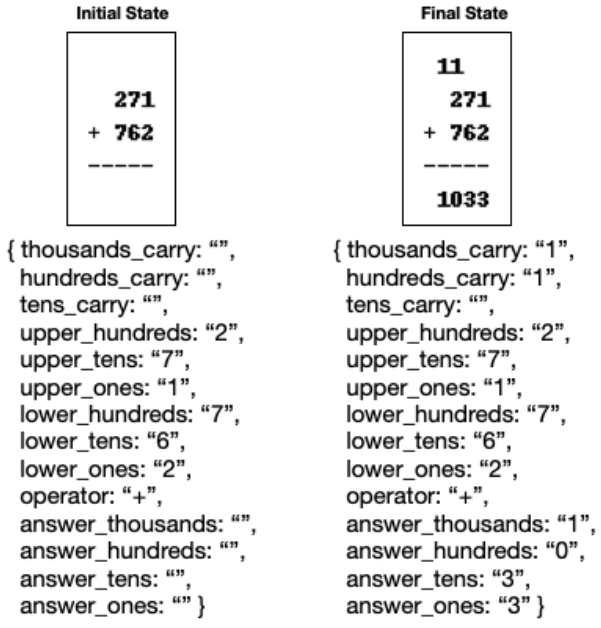


Figure 2: Multicolumn tutor, as rendered within TutorGym with its underlying base feature representation.

underlying tutor. As RL models only learn from feedback on actions, these interactions are mainly used by the Apprentice models, which learn from both examples and feedback.

TutorGym logs agent interactions in DataShop format [12], which is a common educational data format. Outputting data in this format lets us analyze it using the same techniques used to analyze human tutor data. In particular, it lets us conduct learning curve analysis to investigate agents' first attempt correctness as they receive more practice.

We implemented two tutors within TutorGym: a fraction arithmetic tutor [21] and a multicolumn addition tutor [30]. We chose these tutors because they exhibit interesting state and action spaces characteristics that are relevant to our analysis of emerging RL approaches.

3.1 Fraction Arithmetic Tutor

This tutor was used to study both human [21] and agent [16] learning. It presents students with three kinds of fractions problems: addition with the same denominator, addition with different denominators, and multiplication. The students check a box indicating whether they need to convert to common denominators before solving. If the fractions need to be converted, then they input values into the conversion fields. The tutor requires students to convert fractions to common denominators using cross multiplication. If students do not need to convert, then they directly enter values into the final fraction fields. Figure 1 shows the visual representation of the tutor state generated by TutorGym along with the underlying attribute-value representation that it maintains internally. The tutor gives students randomly generated problems where the initial numerators range from 1-15, the initial denominators range from 2-15, and the type of problem can be either addition or multiplication. There is also an "easy" version of the tutor that generates a much smaller range of numbers (numerators range from 1-5 and denominators range from 2-5). The tutoring system also has a done button (not shown) that the agent can select and it can provide worked examples on request.

3.2 Multicolumn Addition Tutor

The second tutor was used in previous research on simulated students [30]. It presents students with two numbers to add, with each digit presented in its own field. To compute the solution, the students needs to add and carry values where necessary. The tutor requires students to enter the answer values right to left, carrying where necessary. The tutor will mark an answer incorrect if they have not yet filled in the answer field to the right or they have not yet carried over a value from the previous column (if required). Figure 2 shows a simple visual output created by TutorGym along with the underlying attribute-value representation. The tutor also has a done button (not shown) and can provide worked example on demand.

4. LEARNING MODELS

4.1 Apprentice Learner

We created three alternative learning models to train within the TutorGym tutors. We built our first model using the Apprentice Learner architecture [16, 30]. From this architecture, we used the an Apprentice model developed in prior work [17]. For each tutor, we provided the apprentice model with background relational knowledge (for augmenting the state description) and primitive operators (for explaining demonstrations). For the fractions tutor, we provided equality knowledge, which adds features to the state description for each pair of fields denoting whether they have equal values. We also provided three primitive operators: copy, add, and multiply, which give the agent the ability to copy, add, and multiply values from the interface.

For the multicolumn tutor, the knowledge was slightly more complicated. We added four relational knowledge operators: add2-ones, add2-tens, add3-ones, and add3-tens. The first two add the values from every pair of fields in the interface and add features to the state denoting the ones component and tens component of each sum. Add3-ones and add3-tens do the same, but for every triplet of fields. This provides the agent with the ability to determine if a column of numbers (either of length two or three) will generate a value that needs to be carried or not. We also added these exact same operators as primitive operators, so the agent can use them to explain and perform the actual steps of computing each column sums and generating the appropriate carry values.

4.2 PPO-Number

A PPO model is defined in terms of its state (input) and action (output) representations. For the fractions and multicolumn tutors, the PPO-Number model makes use of the base state-representations shown in Figures 1 and 2. To convert these representations into a format that is acceptable to PPO (a fixed-length feature vector), we used an approach called one-hot encoding. Under this scheme, every unique attribute-value pair from the state is mapped to a particular feature in a feature vector. If the attribute-value is present in the state, then the feature is a 1, otherwise it is a 0.

Unfortunately, precomputing all possible attribute-values is non-trivial. To address this issue, we created an *online* one-hot encoder that always outputs a vector with a fixed length preselected by the user. Whenever the encoder encounters a new attribute-value, it maps it to a previously unused feature within the vector. After a mapping between an attribute-value and a feature has been made, that feature is only ever used to represent that particular attribute-value. This scheme enables the use of RL approaches that expect fixed-length vectors (PPO) even though the system might encounter a large number of sparse features that are not known in advance. The end result is that states from the fraction and multicolumn tutors are mapped to fixed length feature vectors, where every feature is either a 0 or a 1 (e.g., the initial state from Figure 2 would have a feature for $upper\_tens = 7$ with a value of 1). For the fractions tutor, 2000 features was sufficient to describe states in the standard tutor and 900 features was sufficient to describe the states in the easy tutor (with a smaller set of problems). For the multicolumn tutor, 110 features were sufficient.

Given this state representation, PPO-Number utilizes a multidiscrete output. This type of output has multiple independent discrete action outputs; e.g., in Atari it might have an output for the arrow pad (left, right, up, down) and another output for the action action buttons (A, B, or None). PPO-Number also has two outputs: one that outputs a field to enter a value into (e.g., answer_num) and a second that outputs a number to enter into that field (e.g., 1).

For the fractions tutor, there are eight fields that can be selected for input and there are 450 possible numbers that can be entered into one of these fields (1-450). For the easy version of the tutor, there are only 50 possible numbers (1-50). Taken together, this means that the standard tutor has 3,600 unique actions ($8 \times 450 = 3600$) and the easy tutor has 400 unique actions ($8 \times 50 = 400$). There are slightly less actions in practice because outputs are ignored in certain cases; if the system selects the done or the check_convert fields, than the number component is ignored.

For the multicolumn tutor, there are also eight possible fields that can be selected. Each field represents a single digit, so there are only 10 numbers that can be input into each field (0-9). This yields a total action space of 80 actions ($8 \times 10 = 80$). Similar to fractions, the total is slightly less in practice because the number component of the output is ignored when the done button is selected.

4.3 PPO-Operator

This model uses a different state and action representation from PPO-Number. The representation aims to mirror the representation used by the Apprentice model. We apply the relational knowledge used by Apprentice to augment the base state representation from each tutor. In the fractions tutor, we apply the equality relation to add an additional feature describing which pairs of fields are equal. For the multicolumn tutor, we apply the add2-one, add2-tens, add3-ones, and add3-tens relations to compute the ones and tens values for the sums of every unique pair and triple of values from the tutor fields. We applied the same one-hot encoding approach used for PPO-Number to convert attribute-values into fixed-length feature vectors. We increased the size of the feature vectors to support the combinatorial number of additional relational features (2000 for fractions and 5000 for multicolumn).

The action space is multidiscrete, but the number and type of outputs are slightly different from PPO-Number. For the fractions tutor, the model has four outputs. The first is similar to PPO-Number's selection output, it identifies the field to update with a result. There are eight possible fields that might be updated by an action. The second output corresponds to an operator to apply. The operators are the same as those available to Apprentice: copy, add, or multiply. The remaining two outputs correspond to fields in the interface that provide the two argument for each operator and there are ten possible fields that can be used for either of these arguments. Using this scheme, an agent might choose to update the answer_num field using the add operator, and it might provide the initial_num_left and initial_num_right as arguments. There are 2400 possible unique actions under this representation ($8 \times 3 \times 10 \times 10 = 2400$). However, this number is smaller in practice. If the model chooses to update the done or check_convert fields than the reset of the action outputs are ignored. Additionally, if the model chooses to use the copy operator, than only the first argument is used (the second is ignored).

For the multicolumn tutor there are five outputs instead of four. The first corresponds to a field to update (there are eight possible fields). The second corresponds to the operator to apply. There are five operators corresponding to those used by Apprentice: copy, add2-tens, add2-ones, add3-tens, add3-ones. Finally, there are three argument fields because some of the operators (add3-tens and add3-ones) take three arguments. There are 13 possible options for each argument. With these outputs, there are 87,880 unique actions ($8 \times 5 \times 13 \times 13 \times 13 = 87880$). In practice this number is much smaller because if the done field is updated, then all the other outputs are ignored. Similarly if the copy operator is selected, than only the first argument is used (second and

Model	Domain	# Inputs	# Discrete Outputs
Number	Fractions	2000	8, 450
Number	Fractions-Easy	900	8, 50
Number	Multicolumn	110	8, 10
Operator	Fractions	2000	8, 3, 10, 10
Operator	Multicolumn	5000	8, 5, 13, 13, 13

Table 1: Size of PPO model input/output for each task.

third arguments are ignored). Finally, if the add2-tens or add2-ones operator are selected, than the third argument is ignored. Table 1 shows a summary of the number of inputs and outputs for each model and tutor.

5. SIMULATION STUDY

5.1 Tuning and Training Models

We conducted a simulation study with a factorial design, where every agent (Apprentice, PPO-Number, and PPO-Operator) was trained in each environments (fractions and multicolumn). The hyperparameters used by PPO greatly affect its performance [10] and they must be tuned independently for each model and task. We used Optuna, an open-source hyperparameter optimization framework to automate hyperparameter search [2]. Using Optuna, we ran approximately 100 iterations of hyperparameter tuning for each PPO model and task pair. Tuning one model for one task took approximately 38 hours. The Apprentice model does not have any hyperparameters that need to be tuned.

We trained each model in each environment using the best hyperparameters. We trained Apprentice on 500 fractions problems and 5000 multicolumn problems. These amounts provided enough practice to reach mastery while minimizing unnecessary computation. We trained each PPO model for 1 million steps, which translates into a varying number of problems depending on the amount of incorrect steps. To analyze the simulation logs, we assigned knowledge component labels to each field for each problem type (e.g., answer one's place for multicolumn), computed the first-attempt correctness on each knowledge components for each problem, and plotted this correctness on a log scale with values smoothed using binomial Gaussian additive smoothing (to account for the 0/1 nature of the correctness values).

5.2 Results

See Appendix A for the results of hyperparameter tuning. During tuning, we were unable to get PPO-Number to converge to a correct model in the fractions tutor. We hypothesized this was due to the large number of actions for this model/task. To test this hypothesis, we trained PPO-Number on the easy fractions tutor, which has substantially less actions. PPO-Number converged to correct behavior on this tutor, supporting our hypothesis.

Figures 3 and 4 shows the learning curves for the different models in the two tutor environments. We find that Apprentice converges to mastery after 10 opportunities for each knowledge component in the fractions tutor and 125 in the multicolumn tutor. In contrast, PPO-Number requires over 10,000 opportunities to reach mastery on the *easy* fractions tutor and over 10,000 practice opportunities to reach mastery in the multicolumn tutor. PPO-Operator requires less opportunities (3,000) within the fractions tutor , but never quite reaches mastery within the multicolumn tutor, even after 10,000 opportunities. Even though both PPO-Number and PPO-Operator receive the amount of training steps (1 million), PPO-Operator makes more mistakes per problem and receives less problems as a result.

5.3 Discussion

At least one PPO model was able to achieve mastery in each tutor. PPO-Operator achieved mastery in the fractions tutor and PPO-Number achieved mastery in the multicolumn tutor. This suggests that PPO can generalize from game and robotics tasks to tutor tasks. However, the finding that no single representation is best suggests that the representations must be customized for each task.

PPO-Number was unable to master the standard fractions tutor. We suspect this is due to the single output channel

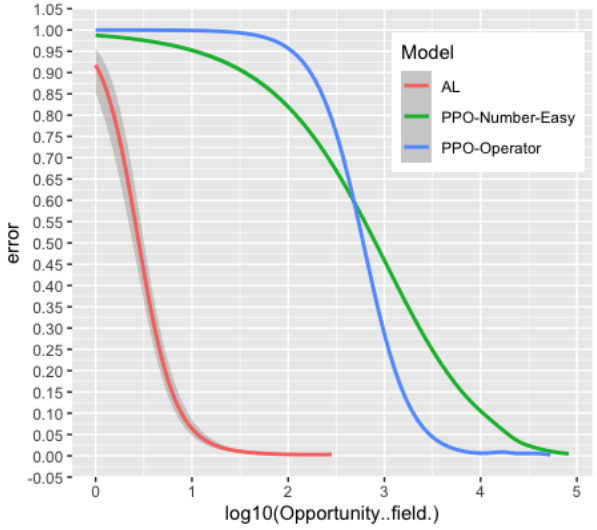


Figure 3: Fraction arithmetic learning curves.

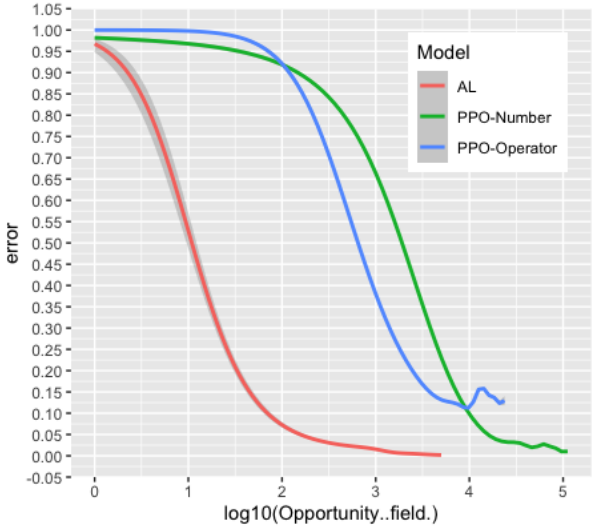


Figure 4: Multicolumn addition learning curves.

with 450 actions. Based on our experience, model performance degrades when the number of actions on one of the multidiscrete outputs gets large. Future work should explore replacing the 450 action output with three outputs: a hundreds digit output (0-4) a tens digit output (0-9) and a ones digit output (0-9). We also found that PPO-Operator was unable to achieve mastery in the multicolumn tutor. It may achieve mastery with more training (e.g., 1.5 million training steps rather 1 million). Assuming this is true, then PPO-Operator, which uses the same relational and operator knowledge as Apprentice seems to be more generally applicable than PPO-Number. Apprentice achieves mastery in both tasks with substantially less practice (thousands of times less), suggesting Apprentice has more efficient learning. Apprentice models have been shown to have similar learning curves to human students [16] for the fractions tutor. This implies that PPO models require substantially more training than human learners.

One limitation of the current study is that PPO may be more efficient when trained with multiple simultaneous environments (e.g., 8 tutors in parallel). Parallel training provides more diversity and improves learning. We tested this idea by training PPO models on 8 parallel environments for both the fractions and multicolumn tutors. We found that parallel PPO required an equivalent amount of practice to achieve mastery as non-parallel PPO; however, parallel PPO took less total wall time to train (e.g., 12 instead of 48 hours). Future work should explore the benefits and trade-offs of parallel training. Also, PPO is an on-policy RL approach, as opposed to an off-policy approach like Deep Q-Networks (DQN) [20]. As such, PPO only trains on the data that is immediately sampled from the environment; it discards old training data because it will cause the model to diverge. In contrast, DQN saves all experiences and continues to train on them over the course of learning. We would have liked to compare PPO to DQN, but DQN does not support multidiscrete action outputs, so could not be evaluated using the current Number and Operator representations. Future work should explore modifications of DQN (or other off-policy models) to see how they perform on these tasks.

6. RELATED WORK

There has been substantial work exploring the use of RL to optimally sequencing students' practice [9]. Unfortunately, this approach requires a large amount of data. One solution is to train models using simulated student data. However, simulated student models are often simplistic and not representative of real student behavior [8]. As a result, sequencing models built from synthetic data typically perform poorly with human students. The RL models we propose might serve as better simulated student models. Adopting a rational analysis perspective [3, 11], we hypothesize that agents that face the same task and processing constraints as humans will have similar behavior; i.e., sequencing models that are best for agents should be best for humans. However, future work is needed to investigate this hypothesis.

A similar parallel hypothesis is that when the task and processing constraints between RL and humans differ, the their behavior is likely to differ. To investigate this idea, Stamper et al. [26] explore differences in human vs. RL expertise for two games: Connect Four and Space Invaders. We view our work as complementary to this research, and future work should compare the behavior of expert models learned in this work to the behavior of human experts.

Simulated students have been used for a wide range of applications including theory testing [16], expert model authoring [17, 30], and teachable agents [18]. However, we are unaware of previously developed simulated students that make use of RL. Some of this prior work aims to model human learning and behavior. In contrast, this work makes little effort to model humans. We view this as a shortcoming of our current study, due to its preliminary nature. Future work should explore how RL approaches, such as those explored here, might be integrated within human-like simulated student models, such as Apprentice.

7. FUTURE WORK

TutorGym and our initial PPO models lay the foundation for a number of novel research directions. One promising directions we hope to explore concerns the use of RL to discover buggy student knowledge. During learning, RL agents make many mistakes. We should explore how these mistakes relate to the kinds of mistakes that humans make. VanLehn [27] investigated the "mind bugs" that human students exhibit in multicolumn arithmetic. Future work should explore how RL bugs compare to human bugs and if RL can support the discovery of bug knowledge for tutor tasks.

8. CONCLUSIONS

We explore the application of the PPO—an emerging RL approach—to educationally relevant tasks. While RL has been successfully applied to learn expert models across many tasks and domains, it has not yet been applied in the context of educationally relevant tasks. To support this exploration, we created TutorGym, a toolkit for interfacing RL models with educational training environments. We created two tutor-based environments within TutorGym: a fraction arithmetic tutor and a multicolumn addition tutor.

We created two PPO models that differ in their state and action representations (PPO-Number and PPO-Operator). For comparison purposes, we created a simulated student model using the Apprentice Architecture that has a similar state and action representation to the PPO-Operator model, but uses different (non-RL) learning mechanisms that are specifically designed to model human learning. We conducted a factorial study that varied the model and task. We found that at least one PPO model is able to achieve mastery within each tutor, suggesting that PPO is applicable to educationally relevant tasks. Despite this success, we found that both PPO models require substantially more training to reach mastery than Apprentice. This suggests that educationally relevant tasks present an interesting use case for the study and advancement of RL research. We also found an interaction between the type of PPO model and the task (PPO-Operator is best for fractions, but PPO-Number is best for multicolumn). This suggests that PPO's representation affects its performance and must be customized specifically for each task.

This work lays the foundation for future research to study and develop RL approaches for educationally relevant tasks. Our hope is that TutorGym and our initial models enable new research into how RL can support human education.

9. REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2004.

[2] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2019.

[3] J. R. Anderson. *The adaptive character of thought*. Psychology Press, 1990.

[4] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[5] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[6] BulletPhysics. Pybullets real-time physics simulation. `https://pybullet.org/wordpress/`, March 2021.

[7] DARPAtv. Alphadogfight trials final event. `https://www.youtube.com/watch?v=NzdhIA2S35w`, August 2020.

[8] S. Doroudi, V. Aleven, and E. Brunskill. Robust evaluation matrix: Towards a more principled offline exploration of instructional policies. In *Proceedings of the ACM Conference on Learning@Scale*, 2017.

[9] S. Doroudi, V. Aleven, and E. Brunskill. Where's the reward? *International Journal of Artificial Intelligence in Education*, 29(4):568–620, 2019.

[10] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proceedings of the Conference on Artificial Intelligence*, volume 32, 2018.

[11] A. Howes, R. L. Lewis, and A. Vera. Rational adaptation under task and processing constraints: Implications for testing theories of cognition and action. *Psychological review*, 116(4):717, 2009.

[12] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the edm community: The pslc datashop. *Handbook of educational data mining*, 43:43–56, 2010.

[13] N. Li, N. Matsuda, W. W. Cohen, and K. R. Koedinger. Integrating representation learning and skill learning in a human-like intelligent agent. *Artificial Intelligence*, 219:67–91, 2015.

[14] N. Li, E. Stampfer, W. Cohen, and K. Koedinger. General and efficient cognitive model discovery using a simulated student. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2013.

[15] C. MacLellan, K. Stowers, and L. Brady. Optimizing human performance using individualized computational models of learning. In *Proceedings of Advances in Cognitive Systems*, 2020.

[16] C. J. Maclellan, E. Harpstead, R. Patel, and K. R. Koedinger. The apprentice learner architecture: Closing the loop between learning theory and educational data. In *Proceedings of International Conference on Educational Data Mining*, 2016.

[17] C. J. MacLellan and K. R. Koedinger. Domain-general tutor authoring with apprentice learner models. *International Journal of Artificial Intelligence in Education*, pages 1–42, 2020.

[18] N. Matsuda, W. Weng, and N. Wall. The effect of metacognitive scaffolding for learning by teaching a teachable agent. *International Journal of Artificial Intelligence in Education*, pages 1–37, 2020.

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[21] R. Patel, R. Liu, and K. R. Koedinger. When to block versus interleave practice? evidence against teaching fraction addition before fraction multiplication. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2016.

[22] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. Stable baselines3. `https://github.com/DLR-RM/stable-baselines3`, 2019.

[23] Robotii, Inc. Mujoco advanced physics simulation. `http://www.mujoco.org/`, March 2021.

[24] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[26] J. Stamper and S. Moore. Exploring teachable humans and teachable agents: Human strategies versus agent policies and the basis of expertise. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 269–274. Springer, 2019.

[27] K. VanLehn. *Mind bugs: The origins of procedural misconceptions*. MIT press, 1990.

[28] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[29] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[30] D. Weitekamp, E. Harpstead, and K. R. Koedinger. An interaction design for machine teaching to develop ai tutors. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2020.

[31] Q. Zhang and C. MacLellan. Going online: A simulated student approach for evaluating knowledge tracing in the context of mastery learning. In *Proceedings of International Conference on Educational Data Mining*, 2021.

	PPO-Number Fractions-Easy	PPO-Operator Fractions	PPO-Number Multicolumn	PPO-Operator Multicolumn
n_steps	1024	256	32	128
batch_size	512	32	32	64
gamma	0.0	0.0	0.0	0.0
learning_rate	4.75e-3	4.56e-5	1.43e-3	7.13e-4
lr_schedule	constant	constant	linear	constant
entropy_coef	6.27e-3	3.28e-2	4.21e-2	2.91e-3
clip_range	0.2	0.1	0.2	0.4
n_epochs	1	10	5	1
gae_lambda	0.98	0.99	0.92	1.0
max_grad_norm	0.8	0.5	0.7	0.3
vf_coef	0.915	0.240	0.401	0.568
net_arch	small	tiny	medium	small
shared_arch	False	True	False	True
activation_fn	tanh	tanh	relu	tanh

Table 2: PPO hyperparameters identified using hyperparameter optimization.

APPENDIX

A. HYPERPARAMETER TUNING

For each tuning trial, Optuna selects hyperparameter from a prior sampling distribution, trains the model using these values, and measures the resulting performance. Within a trial, the PPO model is trained for 350,000 steps. The final model performance is used to update the hyperparameter sampling distribution, so subsequent iterations sample more promising hyperparameters. Optuna also implements a sample pruner, which detects PPO trials that are under performing (e.g., if PPO performance gets worse with training rather than better) and prunes these samples early.

Table 2 shows the hyperparameters that were identified by Optuna for each PPO model and domain. The hyperparameter values are not particularly interpretable, but we report them here so other researchers can replicate our results. It is worth noting that we manually fixed the gamma value at 0.0, since tutoring systems provide immediate reward and future rewards do not need to be factored into decision making. Additionally, the tiny net architecture used a neural network with two layers and 32 nodes per layer, the small network used 64 nodes per layer and the medium network used 128 nodes. If the architecture was shared, then the second layer of the network was shared by both the value and the policy head of the network. However, if they were not shared then there were separate second layers for the value and policy heads. Finally, a constant lr_schedule means that the learning rate is held constant over the course of training, whereas a linear schedule means that learning rate is decreased linearly towards 0 over the course of training.

Do Common Educational Datasets Contain Static Information? A Statistical Study

Théo Barollet
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble France
theo.barollet@inria.fr

Florent Bouchez Tichadou
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble France
florent.bouchez-tichadou@imag.fr

Fabrice Rastello
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble France
fabrice.rastello@inria.fr

ABSTRACT

In Intelligent Tutoring Systems (ITS), methods to choose the next exercise for a student are inspired from generic recommender systems, used, for instance, in online shopping or multimedia recommendation. As such, collaborative filtering, especially matrix factorization, is often included as a part of recommendation algorithms in ITS.

One notable difference in ITS is the rapid evolution of users, who improve their performance, as opposed to multimedia recommendation where preferences are more static. This raises the following question: how reliably can we use matrix factorization, a tool tried and tested in a static environment, in a context where timelines seem to be of importance.

In this article we tried to quantify empirically how much information can be extracted statically from datasets in education versus datasets in multimedia, as the quality of such information is critical to be able to accurately make predictions and recommendations. We found that educational datasets contain less static information compared to multimedia datasets, to the extent that vectors of higher dimensions only marginally increase the precision of the matrix factorization compared to a 1-dimensional characterization. These results show that educational datasets must be used with time information, and warn against the dangers of directly trying to use existing algorithms developed for static datasets.

Keywords

Knowledge tracing, Recommender systems, collaborative filtering, static models, matrix factorization

1. INTRODUCTION

Knowledge tracing tries to model the knowledge of students as they learn, and is a key component of Intelligent Tutoring Systems (ITS). In such systems, the aim is to recommend resources, such as exercises (or "problems"), to students in the most effective way, that is, to recommend resources which correspond to their learning needs. These resources can be of various forms, but in this article we focus solely on recommending new problems for the student to solve. In order to perform any recommendation, we believe that we should be able to predict the outcome of one particular student trying to solve one particular problem; we call this a *(student, problem)* pair. Ideally, we have perfect information for all such *(student, problem)* pairs, whether that information is actual (extracted from observation) or deduced (based on previous observation). This would allow us, for instance, to skip problems that are predicted as being too easy or too hard for a student.

Each *(student, problem)* pair could reflect a level of "difficulty" indicating the student's proficiency. In such a system, one would derive existing difficulty levels from known interactions, for instance through how much time was required for a student to solve a problem, or how many attempts it took to successfully solve it. A "good" system would then predict difficulty levels for interactions that did not happen, possibly with a confidence measure of the outcome prediction. It would also provide an understanding of the structure of the problem set. For example, it would enable the recognition of problems that train similar skills or use similar knowledge, without relying on expert knowledge components that require human expertise.

Historically, the field of knowledge tracing has been independent of recommender systems. With expert knowledge components, one can explicitly measure student proficiency with simple models like Item Response Theory and Bayesian Knowledge Tracing [2]. Using data mining on large datasets, it is possible to relax the knowledge components to be latent features that do not require human experts to partition the domain into explicit student skills [11]. Techniques in educational data mining are inspired from techniques used in collaborative filtering [1], such as factorization methods [20, 18], but also from techniques used in deep learning such as deep knowledge tracing [11, 19, 20, 13].

In this article, we will focus on matrix factorization methods. These are traditionally used in contexts where the available data is not very sensitive to time, for instance movie tastes and shopping habits. In contrast, students learn each time

Théo Barollet, Florent Bouchez-Tichadou and Fabrice Rastello "Do Common Educational Datasets contain Static Information? A Statistical Study". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 310-316. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

they practice and should normally improve with time, so it would make sense to take history into account when analyzing datasets, making predictions and doing structural studies. However, we do not know how much impact this has on the results. The question that we would like to raise here is whether taking history into account is that important or if it is still possible to make good predictions when considering datasets as timeless. Recommending a good problem in terms of teaching is not an easy task, but it is even more difficult when we cannot reliably predict whether the student will succeed or fail, and how long it will take them to do so. In the rest of this article, we will study how the matrix factorization algorithm behaves in three datasets from the educational data mining community compared to one dataset from the traditional collaborative filtering community. We will often deliberately *leave out* chronological information in the educational datasets to see how much information can still be extracted, compared to a traditional dataset.

This article makes two main claims:

- Educational datasets contain much less static information than usual datasets found in multimedia recommendation. Hence, treating educational datasets without any dynamic method should be avoided;
- The little static information they contain amounts to a one-dimensional value per student or problem.

We also propose in Section 3 a pre-processing procedure for educational datasets meant to facilitate prediction, even though we could not find any variable that is accurately predicted among all tested datasets, as well as a filtering procedure to try to find clusters among students and problems that are particularly accurately predicted in Section 5.

2. RELATED WORK

2.1 Data Pre-processing

It is sometimes necessary or advantageous to perform preprocessing of the data before trying to extract information. For example, it was possible to improve the classification error in the MNIST database from 12% to 8%, keeping the same linear classifier, by using deskewing pre-processing [8]. Regarding our knowledge tracing problem, many corrections to the ASSISTment dataset [4] are proposed by Xiong et al. [19]. They are now included in the public dataset that we use later. We also propose some more pre-processing in Section 3.

2.2 Matrix Factorization

Matrix factorization (MF) is a widely used technique in recommender systems, as illustrated by its extgensive usage in the 2009 Netflix Prize Competition [7]. We consider a set U of N *users*, a set I of M *items*, and a set of *ratings* R. These sets are usually given as *records* $(u, i, r_{u,i})$, representing how much $(r_{u,i})$ a given user u likes item i. From these we can build a sparse rating matrix $X \in \mathbf{R}^{N \times M}$. The goal of matrix factorization is to find two matrices $W \in \mathbf{R}^{N \times k}$ and $H \in \mathbf{R}^{M \times k}$ (usually with low rank $k \ll N, M$) such that X is close to $WH^\intercal$. This is an optimization problem written as:

$$\underset{\substack{W \in \mathbf{R}^{N \times k} \\ H \in \mathbf{R}^{M \times k}}}{\operatorname{argmin}} \sum_{(i,j) \in \Omega} (X_{ij} - w_i h_j^\intercal)^2 + \lambda(\|W\|_F^2 + \|H\|_F^2) \quad (1)$$

Where λ is a regularization meta-parameter and $\|.\|_F^2$ is the Frobenius norm [21]. Equation 1 may vary in regularization terms (bias, sparsity penalty...) and can incorporate a loss function between X_{ij} and $w_i h_j^\intercal$. We can now estimate unknown ratings within the product $WH^\intercal$. In other words, we look for signatures for users and items in the same latent space of dimension k (i.e., vectors of rank k), such that the outcome of the user rating an item is close to the dot product of these signatures.

This optimization problem is non-convex in general, but different methods exist [7]. The Alternating Least Square (ALS) method is the most popular method as it converges better than the Stochastic Gradient Descent (SGD) method due to non-convexity. When large-scale data is needed, as ALS is not easy to parallelize, and Coordinate Descent is preferred [6, 21]. In a knowledge tracing setting, users are *students* and items are *problems* [20, 16, 18]. Problems are usually split into smaller components that are the *problem steps*. We will see in subsection 3.1 why we recommend a first regrouping pass in order to work with whole problems.

2.3 Cold Start Problem and Online Settings

The *cold start* problem is a typical problem in recommender systems that corresponds to the initial phase of a "nude" system (no data collected yet). The lack of data makes the prediction accuracy unreliable at that early stage. MF techniques are not designed to tackle the cold start problem but, some extensions seek to solve it partially [22, 10]. As we are not focused on prediction accuracy, we will not consider these extensions in this article but we will try to evaluate when the cold start problem ends in Section 4, that is, when there is enough data for MF to start giving results. Trivedi et al. [17] try to solve a cold start problem in an ITS environment with spectral clustering to help refine raw prediction, but they work on the raw features of datasets without student or item signatures.

Even after the cold start, the system usually benefits from new data in general. This is referred to as online recommendation, and MF is widely studied in such a context [5, 9, 12]. These works consider extremely large datasets, about the order of millions of users and items, but it is still feasible to redo a factorization after adding a few elements, as we will see for instance in Section 4 and 5.

3. FLAT PREDICTION AND AGGREGATION

In this section we studied matrix factorization (MF) on four different datasets, and found that not all datasets were directly usable without some pre-processing, compared to classical datasets. We use the basic version (L2 regularization) of Equation 1 with a fixed rank of 20 (apart from the last section where we measure the impact of rank variation). We use coordinate descent for the optimization [21] because some experiments in sections 4, 5, and 6 require numerous factorizations.

3.1 Educational Datasets and Pre-processing

We will use three common educational datasets for the rest of the article: Algebra I 2006–2007 [14], Bridge to Algebra I 2006–2007 [15] (both of these come from the Cognitive Tutor problem set) and ASSISTment09 (we use the corrected

Table 1: Raw data sets overview

Data set	Users	Problems	Steps	Steps occurring once	Mean samples per step	Samples
Algebra I 2006-2007	1338	5644	418 060	314 198	5.4	2 270 384
Bridge to Algebra I 2006-2007	1146	14 787	202 672	46 935	18.1	3 679 199
ASSISTment09	4217	17725	26 688	3123	13.0	346 660
ML-1M	6040	3706	N/A	N/A	269.9	1 000 209

Table 2: Preprocessed data sets overview

Data set	Users	Problems	Samples	Density	Mean samples per problem	Success percentage
Algebra I 2006-2007	1147	3111	152 709	0.043	49.1	0.79
Bridge to Algebra I 2006-2007	1068	8736	235 147	0.025	26.9	0.91
ASSISTment09	2025	12587	238 746	0.009	19.0	0.98
ML-1M	6040	3706	1 000 209	0.045	269.9	N/A

and collapsed version of the dataset) [4]. All three datasets record scaffolding problem statistics (also called *steps* in Cognitive Tutor datasets – we will use both terms here). For each record (also named *sample*), we extract:

1. A student and a *main problem* ID;
2. A *scaffolding problem* ID;
3. A *timestamp* when a student starts a step and the *duration* to complete the step;
4. If the student succeeded at his first attempt: *Correct-First-Attempt* (CFA);
5. The number of *hints* and *errors* of the student for this step.

To our surprise, these datasets are not very usable without pre-processing in comparison with well-established recommender system datasets like the MovieLens dataset. We will compare most experiments with the ML-1M version of this dataset, which will act as a "control" dataset: multimedia recommendation datasets being the canonical use of matrix factorization for recommender systems. The two main reasons for this poor usability that we try to mitigate with pre-processing are the following:

- The notion of scaffolding problem is not standardized between the datasets and is hard to use as is. Some of them are optional, which makes the number of steps for a main problem vary between students. The step order may also change between users, which makes matching between users more difficult at the problem level.
- There is no guarantee of the minimum number of occurrences for a student or a step. Moreover, many steps are done by a single student across the whole dataset, as seen in Table 1 (especially for the Algebra I dataset, where steps can be generated for a student from a template, and are thus unique. These constitute up to 3 quarters of the steps).

Our first pre-processing pass, which is motivated by the very low number of samples per step on average, corresponds to aggregating all the steps of a common main *(student/problem)* pair together. Aggregating timestamps and durations is straightforward (the beginning of a problem is the beginning of the first step and the total duration is the sum of the steps' durations). To aggregate "Correct-First-Attempt" we take the mean across a *(student/problem)* pair so we obtain a floating point value between 0 and 1 instead of a boolean value.

Simply aggregating hint and error counts by summing them is not satisfactory because ultimately we want to have an idea of how much a student struggled on a problem. Summing these quantities is not sufficient to access some basic information such as "*Has the given student reached the end of the problem or given up?*" This information is not provided in the datasets, so we had to build a proxy variable. To answer this question, we need to know, for a given problem, the number of basic steps it decomposes into. To find this quantity, which we call the *problem size*, we counted for each problem/student pair the number of samples. For a student who succeeded (possibly with hints and intermediate mistakes), the problem size and this number should match. We assumed that for a given problem, the most represented number (among all students) was the actual problem size. We believe that this high representativeness comes from the fact that the ITS providing the datasets give enough hints for most students to reach the end of the problem before giving up. This makes the number of hints and errors valuable information to measure the difficulty of a problem for a given student.

Once we have a boolean proxy indicating success by reaching the end of a problem, we can derive two variables: reaching the end without errors and reaching the end without hints. We can also build a difficulty variable to aggregate the hint and error counts: we sum the two counts with a 0.5 coefficient for hints. We represent failure by assigning a difficulty value of twice the maximum value.

After aggregation, we have six variables (called *target variables* or simply *targets* from now on) of interest for each student/problem interaction: *duration* (0-1 scale value), *difficulty* (0-1 scale value), *correct-first-attempt* (0-1 scale value) and *success-reached* (boolean value), *success-no-error* (boolean value), *success-no-hint* (boolean value). The first three are normalized per problem so that for each problem, the "worst" student gets a value of 0, and the best one a value of 1 (giving rise to what we called above a 0-1 scale value). ML-1M has a single target which is the movie rating (also normalized for comparison). After aggregation, we filter out users who have done fewer than 20 problems and problems that are done fewer than 5 times (same threshold than for ML-1M). Table 2 shows the size of the datasets after pre-processing.

3.2 Influence of Aggregation on Datasets

Figures 1, 2, and 3 report the ability of a factorization to accurately model the different target variables on the four

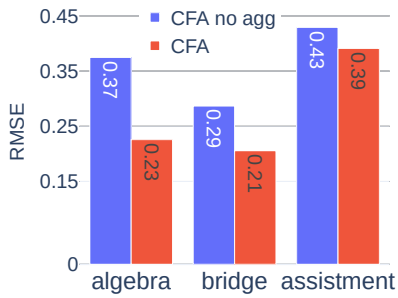


Figure 1: RMSE between no aggregation and aggregation for correct first attempt

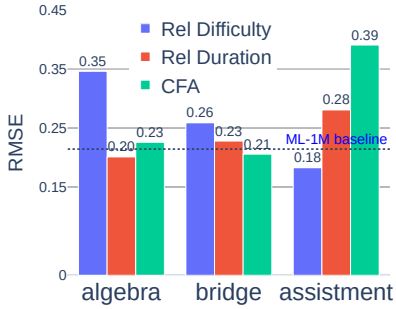


Figure 2: RMSE for duration, correct first attempt and difficulty

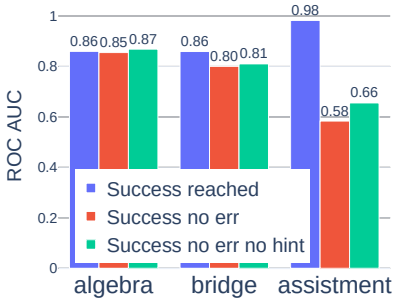


Figure 3: ROC AUC for success and derivatives (higher is better)

datasets and compare the effect of aggregation. For 0-1 scale variables we use the root mean square error (RMSE – the smaller the better) as the error metric, and for boolean variables we use Receiving Operating Curve Area Under Curve (ROC AUC – the closer to 1 the better). For ML-1M, the only possible target variable is the movie rating. We report it in Figure 2 as a horizontal line.

In Figure 1, we see that the aggregation and filter procedure improve the prediction quality of the Cognitive Tutor datasets in a notable way, but only by a small margin for the ASSISTment dataset. We believe that if the pre-processing removed about one third of the problems and half of the students, the density would still be very low compared to the others. In all the remaining experiments, we will use the aggregated versions of the datasets.

It is hard to find any trend regarding the RMSE differences in Figure 2. Variations seem to indicate that different datasets favor different target variables. MF can have about the same prediction capability for educational datasets and multimedia datasets if the target variables are chosen carefully, which suggests that situations call for pre-analyses in order to select the target variable which will be the most accurately predicted.

In Figure 3 we can see that accuracy on success classification is reasonably good. However, we cannot explain the difference in ASSISTment between success-reached and the two other success target variables. This might stem from the aggregation procedure that relies on approximated methods to obtain the number of steps in a problem. We will not do any further experiments on these target variables (which are boolean), as they are barely comparable with the 0-1 scale variable of the ML-1M dataset.

4. ONLINE PREDICTION

In this section, we will try to evaluate the point at which there is enough information to predict reasonably well with MF techniques. This allows the system to stop using whatever bootstrapping technique it was using to solve the cold start problem.

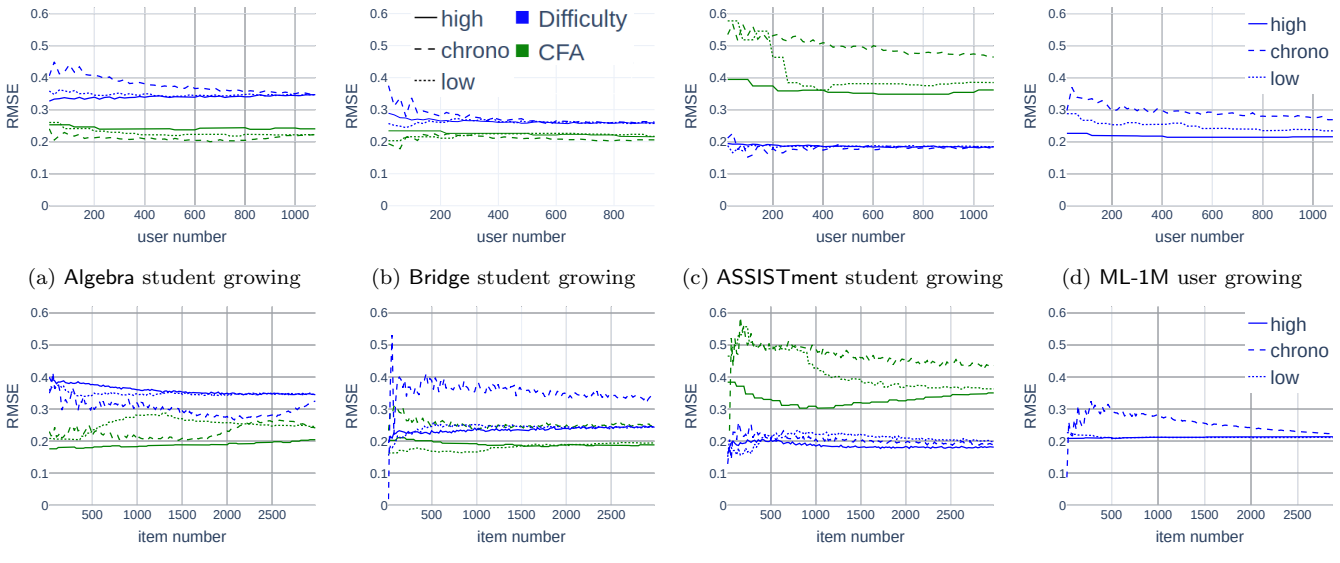
To evaluate this we start by getting either the full student set (and no problem) or the full problem set (and no student). We then progressively add new problems in the first case and new students in the second case, adding 20 new elements at each iteration. At each iteration we redo a full factorization and evaluation as if the system was complete.

We independently measure RMSE of correct-first-attempt and difficulty variables. To evaluate whether the order in which new elements are added makes a difference or not, we considered three orders: (i) elements sorted by their number of occurrences either in decreasing (*high* density first); (ii) or in increasing (*low* density first) order; (iii) and following the chronological order (*chrono*). Only the chronological order makes sense in an online context, but we still use the number of occurrence orders to evaluate whether or not we benefit from a higher density.

We report in Figure 4 the results of this experiment. We can see that for three out of four datasets (not ASSISTment), adding elements by highest density makes the system converge really fast (about 200 elements for Bridge), which was to be expected as those elements carry the most information. For all datasets, adding elements by lowest density, as we might expect, makes the system converge really slowly. We believe that the extremely accurate prediction on some of the curves for the first few iterations of the growing process is due to overfitting (recall that the factorization uses a rank of $k = 20$ in those experiments).

Still, there are some artifacts to these results. In Figure 4e, the previous claims are reversed for difficulty target. Maybe this is a hint that this aggregated variable may not be robust enough on all systems. Our advice is to systematically test target variables on a system to make sure that the ones we choose are consistent and can be trusted.

Finally, we do not observe any "dramatic" drop of the RMSE in curves representing the chronological order that we could clearly label as the "cold start" (although it sometimes takes a few "adds" to stabilize). Of course, the highest accuracy is obtained whenever all the data is used, but this suggests that MF accuracy starts to get close to the maximum early in the process. However, bear in mind that we only evaluated our ability to model existing data (we evaluate on the matrix we factorize), but did not evaluate our ability to predict (by evaluating on the remaining, not factorized, part of the matrix).



(a) Algebra student growing (b) Bridge student growing (c) ASSISTment student growing (d) ML-1M user growing

(e) Algebra problem growing (f) Bridge problem growing (g) ASSISTment problem growing (h) ML-1M item growing

Figure 4: RMSE evolution for student and problem set growing on the four data sets
If not specified the legend is the same as (b).

5. STUDENT OR PROBLEM PREDICTION KERNELS

In this section we search for a subset of students and problems where the prediction is more accurate than on the rest of the dataset. Having such a subset can be of interest in various ways: further analysis like signature clustering might work better on a subset with high accuracy prediction, or this can be a first step towards building a confidence measure for new predictions using a similarity measure with this accurate subset.

Note that we briefly tried some clustering algorithms on the student and problem signatures given by MF, but they were not promising. We will explain in Section 6 how the signatures we can obtain with MF may not be appropriate to such a study.

5.1 Iterative Filtering

We describe an iterative procedure to filter students and problems that have the least accurate predictions.

We alternately remove students and problems: at each iteration we remove the 8% of the considered set that are the least accurately predicted in terms of RMSE (or 15 elements if 8% is lower than 15). We report in Figure 5 and Figure 6 the evolution of the density of the rating matrix and RMSE for the difficulty target variable.

Figure 5 presents the variations in density as we progressively remove students and problems. Interestingly, removing items usually increases the density while removing students decreases it in the three educational datasets, meaning that the students that solved many problems are viewed as "problematic" by the system. This behavior is not observed in the reference ML-1M dataset. Figure 6 confirms the tendency that removing students in the beginning tends to improve prediction accuracy. This result is disturbing as it means that, for the educational datasets, MF prefers less dense matrices with regard to the users, i.e., less information for a given student. This suggests that MF performs best when a problem was done by many students, but when the students have done few problems. What is interesting here is that this scenario is the one that resembles most closely the ML-1M dataset: by having students that did fewer problems, we are indeed eliminating students that likely *progressed* during the experiment, hence whose behavior cannot be represented by a single vector across all their interactions. This is a first solid hint that MF alone does not seem suited to educational datasets, as it shuns chronological subtleties.

6. INFLUENCE OF RANK VARIATION

In this section we repeat the experiments from previous sections with different rank values. In addition to rank $k = 20$ that we already measured, we use rank 5 and a rank of 1. We deliberately choose a rank of 1 to mimic a Whole History Rating (WHR) [3]. Even though it is not an exact correspondence, we believe that the information extracted by a MF with rank 1 can also be extracted by a WHR.

We see in Figures 7, 8, 9 and 10 a clear difference between the educational datasets and ML-1M regarding the influence of ranks. This benefit from rank increase agrees with the intensive use of MF techniques in multimedia recommender systems. However, the benefit of such an increase for educational datasets is almost negligible. This is particularly apparent in Figure 10, where the ML-1M RMSE curves get lower with increasing rank while all other curves are nearly indistinguishable by rank. This shows that, when the chronological information is not used, vectors of size 5 or 20 do not improve accuracy compared to a simple vector of size 1, i.e., a single float. This suggests that we cannot do better than assign a single number to problems and students, which could be interpreted as having a "difficulty"

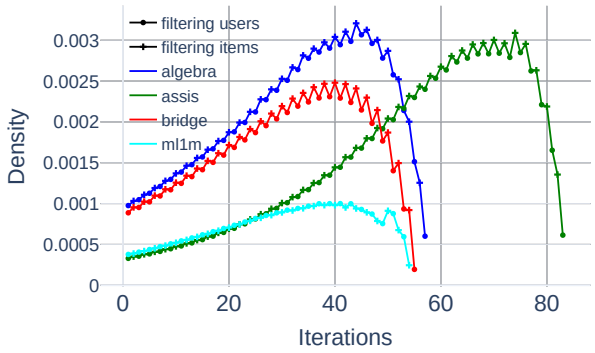


Figure 5: Evolution of density during filtering with target **Difficulty**

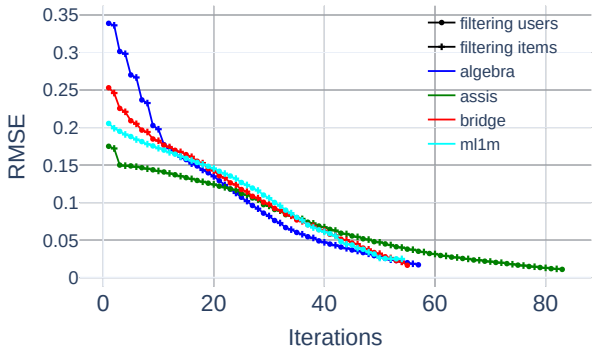


Figure 6: Evolution of RMSE during filtering with target **Difficulty**

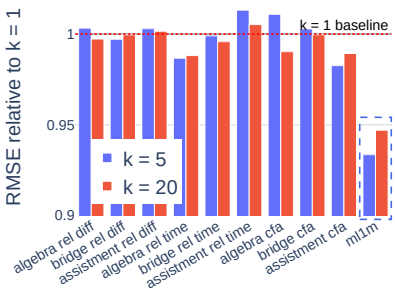


Figure 7: Evolution of flat factorization RMSE depending on rank relatively to k = 1

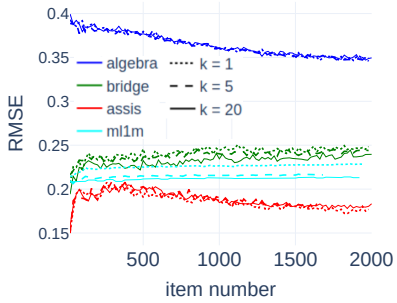


Figure 8: Evolution of RMSE while adding problems with target **Difficulty**

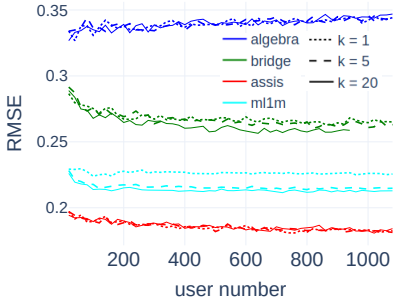
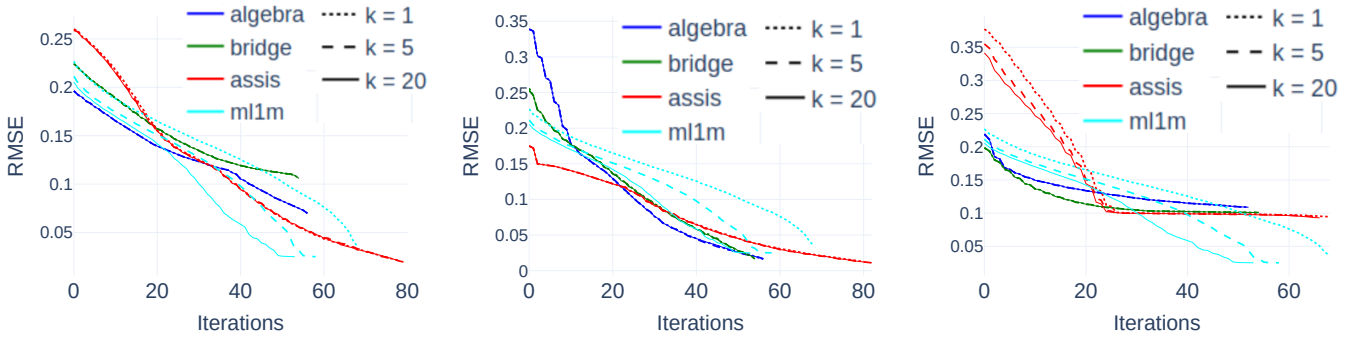


Figure 9: Evolution of RMSE while adding students with target **Difficulty**



(a) Duration (b) Difficulty (c) Correct First Attempt

Figure 10: Evolution of RMSE during filtering with various target

rating for problems and a "skill" rating for students, mimicking a WHR rating system. This is a second strong hint that educational datasets do not have the same structural properties as datasets from multimedia recommenders, and that if we want to extract more information and discover a better characterization of students and problems, it is necessary to consider chronological information.

7. CONCLUSION

We applied preprocessing to common educational datasets to try to improve the accuracy of MF techniques. While these did improve the results, we also showed that when MF techniques from the collaborative filtering community are directly applied, they do not benefit from having ranks higher than 1, meaning that the attribution of a single value to students and problems is about as effective as we can get. This seems to indicate that MF techniques might not be the most efficient model to extract static information from these datasets, or, more probably, that static information is scarce. We believe that this stems from the fact that, unlike users in multimedia recommender systems, students change over time as they are faced with new problems but also from outside interactions not recorded in ITS, hence chronological information needs to be taken into account in order to improve accuracy and make predictions. Still, in the eventual absence of more sophisticated analyses in a recommender system, MF can be used to extract a crude measure of what could be labeled as a level of difficulty of a problem and a level of proficiency or skill of a student.

8. REFERENCES

[1] Y. Bergner, S. Droschler, G. Kortemeyer, S. Rayyan, D. Seaton, and D. E. Pritchard. Model-based collaborative filtering analysis of student response data: Machine-learning item response theory. *International Educational Data Mining Society*, 2012.

[2] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[3] R. Coulom. Whole-history rating: A bayesian rating system for players of time-varying strength. In H. J. van den Herik, X. Xu, Z. Ma, and M. H. M. Winands, editors, *Computers and Games*, pages 113–124, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[4] M. Feng, N. Heffernan, and K. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User Model. User-Adapt. Interact.*, 19:243–266, 08 2009.

[5] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. *ArXiv*, abs/1708.05024, 2016.

[6] C.-J. Hsieh and I. S. Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1064–1072, 2011.

[7] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2290, 1998.

[9] X. Luo, Y. Xia, and Q. Zhu. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems*, 27:271 – 280, 2012.

[10] U. Ocepek, J. Rugelj, and Z. Bosnić. Improving matrix factorization recommendations for examples in cold start. *Expert Systems with Applications*, 42(19):6784–6794, 2015.

[11] C. Piech, J. Spencer, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. *arXiv preprint arXiv:1506.05908*, 2015.

[12] S. Rendle and L. Schmidt-Thieme. Online updating regularized kernel matrix factorization models for large-scale recommender systems. 01 2008.

[13] T. Sergent, F. Bouchet, and T. Carron. Towards temporality-sensitive recurrent neural networks through enriched traces. In A. N. Rafferty, J. Whitehill, C. Romero, and V. Cavalli-Sforza, editors, *Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020, Fully virtual conference, July 10-13, 2020*. International Educational Data Mining Society, 2020.

[14] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Algebra i 2006-2007. development data set from kdd cup 2010 educational data mining challenge. find it at `http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp`, 2010.

[15] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Bridge to algebra 2006-2007. development data set from kdd cup 2010 educational data mining challenge. find it at `http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp`, 2010.

[16] N. Thai-Nghe, L. Drumond, T. Horváth, L. Schmidt-Thieme, et al. Multi-relational factorization models for predicting student performance. In *KDD Workshop on Knowledge Discovery in Educational Data (KDDinED)*, pages 27–40. Citeseer, 2011.

[17] S. Trivedi, Z. A. Pardos, G. N. Sarkozy, and N. T. Heffernan. Co-clustering by bipartite spectral graph partitioning for out-of-tutor prediction. *International Educational Data Mining Society*, 2012.

[18] J.-J. Vie and H. Kashima. Knowledge tracing machines: Factorization machines for knowledge tracing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 750–757, 2019.

[19] X. Xiong, S. Zhao, E. G. Van Inwegen, and J. E. Beck. Going deeper with deep knowledge tracing. *International Educational Data Mining Society*, 2016.

[20] L. Xu and M. A. Davenport. Dynamic knowledge embedding and tracing. In A. N. Rafferty, J. Whitehill, C. Romero, and V. Cavalli-Sforza, editors, *Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020, Fully virtual conference, July 10-13, 2020*. International Educational Data Mining Society, 2020.

[21] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *IEEE International Conference of Data Mining*, 2012.

[22] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 315–324, 2011.

Finding the optimal topic sequence for online courses using SERPs as a Proxy

Sylvio Rüdian
Humboldt-Universität zu Berlin,
Weizenbaum Institute
Berlin, Germany
ruediasy@informatik.hu-berlin.de

Niels Pinkwart
Humboldt-Universität zu Berlin,
Weizenbaum Institute
Berlin, Germany
niels.pinkwart@hu-berlin.de

ABSTRACT

Finding the optimal topic sequence of online courses requires experts with lots of knowledge about taught topics. Having a good order is necessary for a good learning experience. By using educational recommender systems across different platforms we have the problem that the connection to an ontology sometimes does not exist. Thus, the state of the art recommenders can suggest courses with an optimal order within a platform. But on a more global view, a recommendation across different platforms with optimal order is not existing as long as no ontology was defined or courses are not connected to an existing ontology. Nowadays experimental approaches manipulate the learning paths to find the optimum. As this can impact the learning experience of participants, this approach is ethically unacceptable. To overcome this problem, we propose a data-driven approach using the search engine result pages (SERPs) of Google. In our experiment, we used pair-wise search queries to get access to web pages, those 38.000 texts were used to test some NLP metrics. 10 different metrics were examined to create an optimal order that was compared to the optimal sequence defined by experts. We observed that the Gunning Fog Index is a good estimator to determine the optimal order within a cluster of topics.

Keywords

Course Sequencing, educational recommender system, web search, adaptive courseware, personalization.

1. INTRODUCTION

Providing the optimal sequence of topics in online courses is of high interest because it influences the learning outcome as well as motivation. Lots of MOOCs are existing, but in which order they should be done is defined by experts and this is a time-consuming procedure. Large-scale educational recommender systems [1] suggest online courses across different platforms. Creating an optimal sequence based on an ontology is an easy solution as an ontology includes the optimal order, defined by human experts. This can be done within single platforms, but an ontology across different courses across several platforms is not existing. McCrae et al. [2] state that it "is difficult to link to ontologies". The willingness to create a connection of own online courses to an existing public ontology is low as this is expensive due to manual work on the one hand but can also result in course recommendations of other suppliers on the other hand, which does not meet the interests of the suppliers.

The optimal sequence is missing in recommender systems as long as no manually created large-scale ontology or optimal sequence exists. Recommender systems only provide a ranking based on how well the suggested courses fit into the user's learning situation. There are existing approaches, e.g. linked data to create a structured semantic web [3]. Their idea is to create a network that contains the meaning of the data. But there is the problem that the semantic web is limited to specific domains. If the networks have not been created for the topics that we need, we cannot use them. Besides, the structure in the semantic web is designed to understand relationships between objects, not whether there is a dependency from the educational perspective. Further on, there is the problem that topics for online courses often consist of multiple words to describe the topic or concept. Finding the correct corresponding concept within the semantic web can be challenging.

Having an optimal order of online courses is of high interest in online education as many topics require the knowledge of subtopics. Knowledge dependencies can be modeled by experts manually on the one hand, but this is a cost-intense procedure that requires lots of knowledge about the taught topics and provided courses as well. On the other hand, the world wide web is full of contents of different quality. Every topic that can be taught can be found there, but the contents of web pages are still not used for topic sequencing in education. Crawlers get access to all the texts and companies like Google define an order of pages related to a search query. Within a search engine, we get access to all pages that they define to satisfy the user intent [4]. Using this large number of pages for each topic could be beneficial in creating optimal topic sequences for online courses.

An optimal order is very important for a good learning experience in online courses. We define an optimal order as the sequence of course topics where each topic should be taught when all pre-requirements are fulfilled based on the previous courses. As long as topics are taught where the requirements are missing, the dropout rate will be high. Using courseware (single parts of a course) [5] to generate a new online course it is important to have an optimal order. Otherwise, the participant cannot understand the topic because of missing knowledge. The same problem exists in AI-generated learning paths of online courses, which must be consistent according to the fundamental didactical method of starting teaching basics, not with specialized knowledge.

Sylvio Rüdian and Niels Pinkwart "Finding the optimal topic sequence for online courses using SERPs as a Proxy". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 317-322. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Observing the world wide web, we can get a variety of texts on any topic. We want to use this already existing large set of pages to find the optimal topic order of online courses with an experimental approach. To access all the pages with corresponding texts that we need, we use the search engine Google, especially the Search Engine Result Pages (SERPs) [6]. It is known that Google uses the semantic web in the background, depending on the search query, which helps to overcome the challenge to find the optimal corresponding concept of the semantic web using the search engine as a proxy [7]. This is beneficial for the case that for specific domains no linked data is existing – as the search engine tries to provide related resources, even if they are misspelled and the search engine can give results for queries that they have never seen before.

Using topics as keywords results in a list of pages that satisfy the user intent according to Google [4]. This can be used as a base for having access to different features for ordering course topics. SERPs help to understand the popularity of how many pages are indexed by the search engine, which could be an indicator for good sequencing as less specialized topics are existing compared to general basic topics. Besides, having a lookup for two topics in one search, we get pages that contain both keywords, those frequency or deviation could be an indicator for finding the optimal sequence. If we observe online courses, then we usually have an increasing difficulty level. Using the complexity of texts could help to estimate the optimal order based on the difficulty.

In this paper, we concentrate on the three research questions:

1) Is the SERP popularity of all topics a good indicator to find an optimal topic sequence for online courses?

2) Is the topic frequency of page texts that are listed within the SERPs an estimator to determine the optimal topic sequence in online courses?

3) Does ordering topics' texts by text difficulty metrics result in a sequence that is appropriate to be used as a sequence in online courses?

2. RELATED WORK

Brusilovsky et al. [5] define this problem as "sequencing of lessons" where each lesson is connected to a topic. This contains numerous chunks of educational material, ranging from videos and texts to different interactive tasks. The authors use a domain concept structure, that is stored independently from teaching materials. Each concept needs to be linked to the teaching material. It has the advantage of being able to use the courseware to generate a personalized online course according to the interests and knowledge gaps of a learner. This approach is comparable to using an ontology that needs to be defined by experts, based on rules and graph representation. It is the fundamental model to define an optimal sequence of online courses but requires the creation of the ontology by experts.

S. Fischer [8] uses an ontology knowledge base, namely a "knowledge library" to create an optimal course sequencing. Therefore, they use modularized media content as courseware together with metadata that describes the link to the ontology model. With that, they have access to a taxonomy that can be used to create a good ordering of topics as well as generating questions with right and wrong answers (depending on the granularity of the ontology). The modular resources can be used to generate courses, according to the knowledge gaps of learners.

Xu et al. [9] propose to learn from users providing specific course sequences for testing and use their performance to create an optimal sequence for new users. While this approach works it has the disadvantage that it requires real test users which may perform badly within the scenario. Doing this in a field study is acceptable but using real students is not sustainable from an ethical point of view. We want to emphasize that we do not want to use this experimental user behavior data as this is ethically not acceptable.

Cucuringu et al. [10] used already captured student participation in courses to create pair-wise comparisons using ranking aggregation to create a global ranking. This ranking proposes an order of how courses should be taken by students. One major problem is incomplete data as some pairings are not existing for a comparison.

S. Morsy [11] states that a global ranking of online courses cannot be used for personalized recommendations. But having a global ranking can be helpful to determine which courses should be done in which order. Combining this knowledge with personalized courses or topic recommendations is helpful as the course dependencies (e.g. what knowledge is necessary to understand a topic) are the same for personalized recommendations, which are filtered by topics/concepts that the learner is already aware of. Thus having a global ranking can be beneficial for personalization as well.

Using the information of chosen courses by students and their performance is a good way to determine an optimal course sequence. A major limitation with that approach is the limitation of data and to have access to chosen courses and the resulting performance. This approach does not comply with the GDPR as the information on whether students passed or failed an exam is classified to be sensitive personal data, that cannot be accessed for course sequencing in general [12]. Thus, their application does not work in a real-world scenario in the EU. Based on the limitations of being dependent on user performance or manually created ontologies, we propose a new methodology to create an optimal order of online courses, based on their topic.

3. METHODOLOGY

As we learned from Rüdian et al. [13]: Even if experts are scoring the same results of educational tasks, their scores vary among each other. If we observe the order of topics, then we know that there is not always a perfect solution regarding the whole sequence because of ambiguous expert opinions. In the pre-study, four experts (AI instructors) had the task to create the optimal order of 20 AI-related topics to be taught within online courses. We used the following topics: *neural networks, voice recognition, chatbots, Linux, data visualization, Python, statistic basics, part-of-speech tagging, LSTM, data preparation, deep learning, TensorFlow, object recognition, Naïve Bayes, natural language processing, ethical principles, clustering, reinforcement learning, cross-validation, and regression*. The resulting sequences are then used to make a pair-wise comparison to understand the overlap across instructors and to see where we have a high overlap. The pair-wise sequence score S is defined as followed: For every topic A and B of the expert sequence with $A \neq B$ and every topic C and D of the sequence derived by the algorithms or another expert with $C \neq D$ we count all hits where $(A < B \; and \; C < D)$ or $(A > B \; and \; C > D)$. Thus, the topics have the same order within both sequences. This number is divided by the number of possible combinations, defined as S.

Some topics have dependencies; e.g. neural networks should be introduced before teaching LSTM or natural language processing should be taught before starting with part-of-speech tagging; others do not have strong relations and can be taught somewhere, e.g. ethical principles or Linux.

The idea of the main study is to compare the sequences created by instructors with algorithmic ones. Therefore, it is a good fundament

to measure tutors' decisions among each other first to define accuracy as a gold standard that we want to achieve with our methodology. Thus, we need a realistic generalizable accuracy that we should achieve instead of over-optimizing our approach with high accuracy that is the optimum for a sequence of one expert only. Besides, the pre-study identifies the optimal order of the topic subsets that are the same for all experts. The order of these topic subsets will be compared to the order that we get from our algorithms to see how well the algorithms perform in a real-world scenario.

Our approach is to use the Search Engine Result Pages of Google (SERPs) and we derive different metrics based on the results. A search engine can be used to find web pages that are related to given keywords. One of the main purposes of the search engine Google is to satisfy the user intend by providing a list of web pages that are related to the search query [4]. Thus, using it allows us to get pages that have a high authority according to the Google ranking algorithm, which is, according to them, a metric of high quality. We use this list of pages with our topics as search queries to understand the popularity, the number of user searches, the complexity of topics, and which topics have a semantic connection. These metrics are then used to create a sequence, based on a linear order of the observed data. These sequences are compared to the experts' ones to understand whether there is a connection between our metrics and the optimal order, defined by experts.

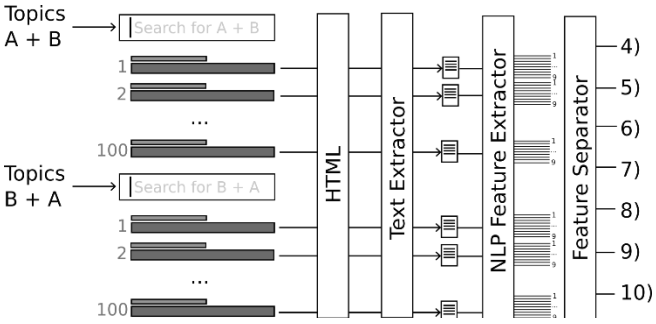


Figure 1. Pipeline for pair-wise search, extraction, and separation of features.

The approach of using a search engine as a basis has the advantage that we do not need to do experiments with students where they may be badly influenced due to bad testing sequences. Thus we are independent and can use our approach on a larger scale. That makes our approach more practically usable. We use different data as a basis, use them to rank our 20 topics, and compare the order with the instructors' ones. Our approaches are the following:

1) We use the number of topic results that are estimated by the search engine by searching for every keyword separately.

2) We use the number of topic results pair-wise keyword combinations and observe the number of estimated results.

3) We use the keyword search estimator and rank our topics according to the estimated search amount.

4) We use the first 100 results of all pair-wise keyword combinations and count how often both keywords within the 100 listed pages exist.

5) We use the first 100 result pages as in 4), search for both keywords on the pages, and summarize how often each keyword occurs at first in the text.

We use the 100 result page texts and apply three algorithms to estimate the text complexity, namely 6) Flesch-Reading-Ease (FRE) [14], 7) RIX [15], and 8) Gunning Fog Index (GFI) [16]. Then we use the 100 result page texts with basic NLP metrics: 9) The type-token ratio (TTR) and 10) the number of words per sentence (NoW). We assume that observing how many pages are existing in combination helps to identify topics that have a semantic connection. Using the information on how many pages are existing gives hints about the popularity (1,3), where for complex topics mostly less content exist than for basics. Observing the complexity of the contents (4-7) could help to identify the difficulty level of topics to find the optimal sequence. Figure 1 visualizes the method for 4)-10) to get features based on a pair-wise topic search.

The Flesch-Reading-Ease Index is based on the "Standard Text Lessons in Reading" [17] and is calculated from the average sentence and word length [14]. The main idea of the Gunning Fog Index is to reduce the complexity in newspapers as a kind of warning system for authors that texts are not "unnecessary complex". Therefore, the author uses the sentence lengths, the number of syllables, easy words, and hyphenated words to estimate the complexity of a text [18]. The "Regensburger Index" (RIX) uses difficulty parameters like passive, sentence complexity, and predications to derive the complexity [15]. All approaches differ in the selection of features that are used to create the indexes.

Finally, we use a random forest regressor [19] to predict the pair-wise sequence, using the data of 4)-10) to estimate the feature importance to support our findings. To get all the data, including the SERPs, all pages, and the estimated search amount, we use a commercial web crawler for SERPs[1]. This is necessary as the pair-wise lookup of 20 keywords results in $20 * 20 - 20 = 380$ searches, where we need to download 100 web pages each, resulting in 38.000 files. A simple crawler that we used in our lab before, was banned after 20 crawls, thus using a commercial one is the most efficient option.

Each data source 1) – 10) is then used to create a ranking of topics, based on their linear order. These sequences are compared with the expert ones to find the optimal feature that can be used in a real-world setting. To compare the sequences of the experts with the algorithmic ones, we use a pair-wise topic comparison to test whether the order is the same in both sequences and summarize the hits. Thus we can compute the overlap that represents the accuracy in our experiments.

4. RESULTS

The overlaps across the expert sequences range from 0.6 to 0.8 (Table 1). Thus we have an orientation of the resulting overlap that can be achieved with our approach at maximum. While the overall sequences defined by experts are partly different, we identified some partial sequences that are identical across all expert-based rankings and use them as ground truth. We detected some matching sequences of topics:

A = ["data preparation" → "data visualization" → "clustering"], B = ["neural networks" → "deep learning" → "LSTM"], and C = ["natural language processing" → "part-of-speech tagging" → "voice recognition" → "chatbots"],

[1] https://seorld.com/crawler

where [A → B] means that topic A needs to be explained before topic B. This makes sense as each topic mostly requires knowledge of the previous one(s), e.g. "neural networks" have to be introduced first and after that, "LSTM" can be explained. We use the three clusters (A, B, and C) to visualize whether our rankings make sense in a real-world scenario as the overlap of sequences defined by a number only is too abstract. All in all, in our pre-study we can conclude that we identified three clusters using sequences of four experts.

Table 1. Pair-wise sequence overlaps of 4 AI experts.

	Expert 1	Expert 2	Expert 3	Expert 4
Expert 1	-	.60	.65	.80
Expert 2	-	-	.65	.65
Expert 3	-	-	-	.75
Expert 4	-	-	-	-

Then we used all the different data points that we got from the crawler separately and created a sequence based on their linear order. Table 2 shows all the results of our experiments. We calculated the pair-wise overlap to compare estimated sequences with the expert ones. Also, we tested whether our partial sequences of the topic sets in A, B, and C have the same order as defined by our experts.

Observing 1) - 3) we can answer the first research question as these metrics represent the popularity of topics within the SERPs. The ordered list of topic pages is not a good indicator to find an optimal topic sequence for online courses. Thus, popularity is not a good indicator of course sequencing.

Table 2. Overlap of sequences with four experts (E1…E4) and the information on whether the orders of our clusters A, B, and C are the same as defined by experts.

Approach	E1	E2	E3	E4	A	B	C
1)	.55	.40	.45	.50	No	No	No
2)	.60	.50	.40	.50	Yes	No	No
3)	.45	.45	.40	.53	No	No	No
4)	.53	.63	.53	.53	No	No	No
5)	.58	.53	.53	.40	No	Yes	No
6) FRE	.35	.50	.55	.50	No	No	No
7) RIX	.50	.50	.50	.45	No	Yes	No
8) GFI	**.55**	**.65**	**.60**	**.60**	**Yes**	**Yes**	**Yes**
9) TTR	.45	.40	.40	.40	No	No	No
10) NoW	.50	.50	.50	.50	No	No	No

Observing the pair-wise searches in 4) and 5) we can conclude, that topic frequency within the related texts is also not a good indicator to get an optimal sequence of topics, which answers the second research question. We limited the search to exact matches. Further, using n-grams or other methods to detect variants could be beneficial.

We identified the Gunning Fog Index as an estimator to create an optimal order. This answers our third research question. Using this metric for text complexity is the most robust feature to create a good sequence of topics in our experiment. Also, the order we got from our clusters is the same as in the sequence that we got by using the GFI. This is very important for a practical educational environment as the orders of topics that have a taxonomy with knowledge dependencies need to be done correctly. The overlap with the expert sequences ranges from 0.55 to 0.65, which is acceptable as the overlap of sequences across experts was in the range of 0.6 and 0.8. The remaining text complexity metrics (6, 7, 9, 10) are not as robust as the GFI.

To get more insights into the importance of the identified predictor, we use the random forest regressor [19] as we investigate linear

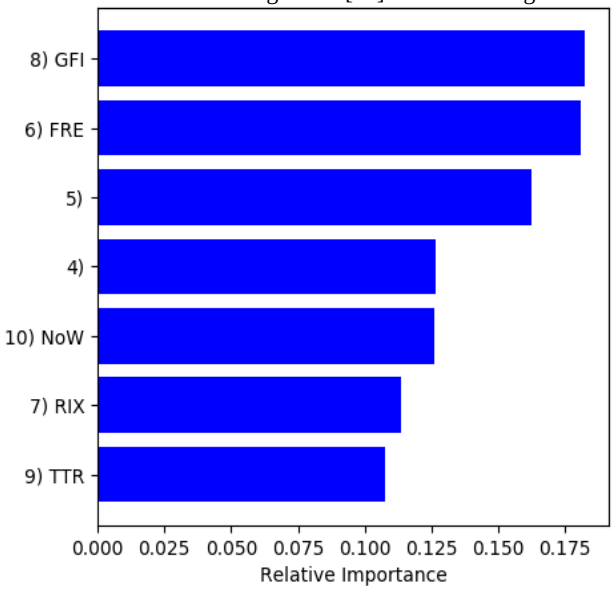


Figure 2. Relative importance of features according to the random forest regressor.

features only and – for future work – we want to identify features of high importance that also work for non-linear dependencies to predict the optimal sequence. Therefore, we use all pair-wise approaches (4-10) to train a decision tree using the random forest regressor. As prediction target, we used all pairings orderings (e.g. topic "neural networks" needs to be taught before topic "LSTM"). This is a classical approach to predict the ordering of items, based on different features. Figure 2 displays the relative importance of features that we got. The most relevant feature is the Gunning Fog Index (GFI), which performs best in our experiments as well.

5. DISCUSSION

Automated analysis of the pair-wise SERPs and the text complexity using the GFI can help to assist instructors during planning course sequences. From an ethical point, doing experiments with students is not justifiable as it could corrupt the learning outcome as a bad implication. As our approach is independent of experimenting with users, this method can be applied on a large scale. Combining this approach with educational recommender systems, we can provide a sequence of topics, based on the topic set that we get from the recommender system, even if no ontology is defined in the background. Using the text complexity helps to start with topics that can be explained more easily than the following ones. Having automatic composed online courses based on courseware, it can be beneficial to use the third party data of SERPs to find an optimal order. This is an important step to create personalized online

courses that are adaptive to the knowledge level, where no pre-defined ontology exists.

There are various fields of application where we can use our approach. This method can be used for planning lecture sequences at school or university, based on the complexity of taught topics. It is the same in preparing new lectures, based on existing learning material, that can be composed in an optimal order. Besides, the curricula at universities could be optimized, where students participate in courses of different universities. Having a recommendation for a good order on which courses should be visited at which point of time is beneficial.

From a practical point of view, it is important to note that the number of searches, while using a commercial crawler, is a cost factor. If $n = "number\ of\ topics"$, then the number of searches $C = n^2 - n$, having pair-wise searches $A + B$ and $B + A$ (with A and B being topics of the list). This is necessary as the SERP list of $A + B$ is not the same as $B + A$. Also, the search query $A + B$ returns 100 pages that need to be crawled to get the texts. In our experiment, this results in 16Gb of data, having 38.000 texts of 20 topics with pair-wise searches, where the metric has to be derived for each text. The required storage grows exponentially with the number of topics. The length of the resulting topic list of educational recommender systems can be limited in general, thus it is not a problem, but it is important to limit the list first, before finding the optimal order to avoid the need for large storage and high computational capacities. Besides, using the first 10 results of the SERPs instead of 100 reduces the crawling budget as well as computing time, but makes the approach less robust.

In our experiment, we conclude that commercial popularity and the estimated search amount are no indicators for a good topic sequence. Independently from the intention of the paper, using popularity is a helpful metric to get insights into trends about what people are searching for. Online course suppliers can use this information to create online courses for a large audience, those sizes can be estimated with the search popularity. As data-driven approaches, e.g. AI-related decisions require lots of participants, offering online courses that are of high interest can help to get the required number of participants to have enough training data for AI methods. From the researchers' perspective having popular courses is of high interest to obtain AI decisions with a high statistical significance. Sources like the Semantic Web do not provide this additional information.

As this is ongoing research, the next step is to create a comparison of the identified cluster sequences with sequences that can be derived using the semantic web as proposed by Toman & Weddell [20]. This real-world experiment can show the applicableness in the field of education. If this method results in similar sequences, we recommend using an already existing semantic network and in case of missing concepts, we can use our method as a fallback.

Observing the overlap of expert sequences, we can see that they are quite diverse. Finding an "optimum in education" is mostly a trade-off between different opinions of experts. We used the sequences to detect partial sequences that are similar across all experts. In the future, all topics should have a description of the taught contents to reduce the variety of sequences. Examining the detected partial sequences, we can see that these topics have a semantic connection and some topics have knowledge dependencies. In a future scenario, we recommend finding clusters of topics first and then use a text complexity metric like the GFI to get the optimal order. Otherwise, there might be a switch of topics, those order is good while looking at the complexity only, but could be confusing on a more global view. From the didactical perspective, switching between different topics in the learning path that have little semantic coherence is not recommended.

In this paper, we focused on AI-related topics to present our research at an early stage. It is of high interest to compare our approach to data from another domain. We assume that the GFI as a complexity metric can be used as an indicator for a useful order as well. But it is important to note, it remains possible that GFI randomly happened to give a good result. Thus extending the experiment to different domains is necessary to give a final and scalable recommendation.

Besides, we assumed that using text difficulty metrics will result in nearly the same order as their task is identical. Observing the results, we can see that there are major differences in the resulting order. The GFI is used to estimate how many years of formal education the reader needs to understand the text on the first reading [16]. In our case, it was the best and most practical metric. Looking at Figure 2, the Flesch-Reading-Ease is also of high importance but failed to create the optimal order of our three clusters (Table 2). Comparing the GFI with FRE, both metrics are based on syntactical features. The GFI is enriched with contextual features like "easy words". This enrichment could be a reason why this index works best in our experiment. Besides, other textual metrics need to be taken into account for testing. Semantical features could be used as well as text entailment. In the future, combining these metrics can be beneficial, e.g. at training a neural network with all metrics to use non-linear dependencies, that were not examined in this paper yet. Textual metrics must be used carefully as they are "just" formulas for judging the complexity of texts [21]. The methods cannot be used to judge the appropriateness of contents or whether the content is correct. Thus, selecting learning material of high quality is important and the metrics are not useful in the selection process.

The proposed approach depends on the SERPs of Google. Having a high fluctuation of rankings within the SERPs could change the feature's importance. As Google regularly updates their algorithms within a core update twice a year, rankings may change [22]. As we use the first 100 results we assume that the approach is robust because there are only minor changes if we consider the set of the first 100 pages. It is debatable that high-ranking Google results contain web pages of high authority, it can be discussed whether the first 100 resulting pages are a good resource for educational purposes and whether they are trustworthy. Instead, they are likely to be optimized for search engines, e.g. by search engine optimizers that create contents with ingoing links of high authority web pages aiming to have high rankings. There is the problem, that often texts of competitors are re-written for new pages to rank for similar terms. Thus, many texts with similar contents can be found. Besides, the SERP came from multiple contributors, they may include low-quality texts from commercial sources and web pages that block search engines are systematically excluded.

We use Google as a proxy to get access to the web pages that contain the texts that we are working with. The same can be done with other search engines. Alternatively, being limited to resources those contents are created by editors of publishing houses for education may be biased as the complexity of texts also depends on the writing style of authors. Using a resource like the first 100 texts results in a more robust view to avoid this bias due to averaged data. It can be discussed whether Google is a good source for characterizing academic terms because SERPs might be too inclusive and therefore noisy. Based on our experiment we could see that a text difficulty average of the gathered data can be a good indicator. Whether this is the case, in general, has to be examined

in further experiments. Besides, we could use educational materials from publishing houses. In general, they are not publicly accessible, which increases the costs if we want to use them. Further research can examine whether resources like Wikipedia or Web of Science can be used with similar metrics to determine the optimal order.

Limiting the approach to the header of courses could generally lead to wrong conclusions if the courses do not cover the topic that was given in the headline. In our experiments, we used topics as keywords only. Using the course description or the course(ware) content itself to obtain more rich information for having richer keywords could be beneficial, that will be addressed in further experiments. Besides, we did not consider synonyms, which should be observed in future studies because using different words (even synonyms) results in different SERPs.

6. CONCLUSION

In this paper, we propose different strategies to use texts that we got using a search engine to find the optimal order of online course topics. The pre-study has shown that the optimal topic sequences differ among experts. But we can also observe that there are partial topics that have the same order in all expert sequences. We identified them to define a gold standard and to check for the practical usefulness. The sequences derived by our approaches were compared to the expert ones and the order of the partial topics. The commercial popularity, that can be derived by searches in search engines is not an indicator of a good topic sequence. Searching for pair-wise topics and comparing the text complexity of the SERPs' web pages' texts can be used as an indicator for creating a plausible order of taught topics within online courses.

We identified the Gunning Fox Index as the most robust metric for topic sequencing. We can conclude that this feature helps to find the optimal sequence for automatic composed online courses to personalize them ethically without using students giving them randomized learning paths that could impair their learning experience as well as their learning outcome.

7. ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry of Education and Research (BMBF), grant number 16DII127 (Weizenbaum-Institute). The responsibility for the content of this publication remains with the authors.

8. REFERENCES

[1] N. Manouselis, H. Drachsler, K. Verbert and E. Duval, "Recommender systems for learning," Springer Science & Business Media, 2012.

[2] J. McCrae, D. Spohr and P. Cimiano, "Linking Lexical Resources and Ontologies on theSemantic Web with lemon," in *Extended Semantic Web Conference*, Springer, 2011.

[3] P. Jain, P. Hitzler, P. Z. Yeh, K. Verma and A. P. Sheth, "Linked Data is Merely More Data," in *D. Brickley; V. K. Chaudhri;H. Halpin; and D. McGuinness, editors, Linked Data Meets Artificial Intelligence*, Menlo Park,CA, AAAI Press, 2010, p. 82–86.

[4] W. Gross, T. McGovern and R. Sturtevant, "Search engine using user intent". USA Patent US20060064411A1, 2005.

[5] P. Brusilovsky and J. Vassileva, "Course sequencing techniques for large-scale web-based education," in *Int. J. Cont. Engineering Education and Lifelong Learning, Vol. 13, Nos.1/2*, 2003, pp. 73-94.

[6] D. Kelly and L. Azzopardi, "How many results per page?: A Study of SERP Size, Search Behavior and User Experience," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, p. 183–192.

[7] T. Steiner, R. Troncy and M. Hausenblas, "How Google is using Linked Data Today andVision For Tomorrow," in *Proceedings of Linked Data in the Future Internet at the Future Internet Assembly*, Ghent, 2010.

[8] S. Fischer, "Course and Exercise Sequencing Using Metadata in Adaptive Hypermedia Learning Systems," in *ACM Journal of Educational Resources in Computing, Vol. 1, No. 1*, ACM, 2001, pp. Article #3, 21 pages.

[9] J. Xu, T. Xing and M. v. d. Schaar, "Personalized Course Sequence Recommendations," IEEE, 2016.

[10] M. Cucuringu, C. Z. Marshak, D. Montag and P. Rombach, "Rank Aggregation for Course Sequence Discovery," 2017.

[11] S. Morsy, "Learning Course Sequencing for Course Recommendation," 2018.

[12] EU, "Processing of special categories of personal data," in *Art. 9 GDPR*, 2018.

[13] S. Rüdian, J. Quandt, K. Hahn and N. Pinkwart, "Automatic Feedback for Open Writing Tasks: Is this text appropriate for this lecture?," in *DELFI 2020 - Die 18. Fachtagung Bildungstechnologien der Gesellschaft für Informatik e.V.*, 2020, pp. 265-276.

[14] T. v. d. Brück and J. Leveling, "Parameter Learning for a Readability Checking Tool," DOI 10.5281/zenodo.2552414, 2019.

[15] J. Wild and M. Pissarek, "RATTE - Regensburger Analysetool für Texte," Universität Regensburg, 2016, pp. 1-12.

[16] R. Gunning, "The fog index after twenty years," in *Journal of Business Communication 6.2*, 1969, pp. 3-13.

[17] F. R, "A new readability yardstick," in *J. Appl Psycho 1* , 1948.

[18] R. Gunning, The Technique of Clear Writing, McGraw-Hill, 1952.

[19] L. Breimann, "Random Forests," in *Machine Learning 45*, 2001, p. 5–32.

[20] D. Toman and G. Weddell, "On Order Dependencies for the Semantic Web," in *Conceptual Modeling*, 2007, pp. 293-306.

[21] J. W.Pichert and P. Elam, "Readability formulas may mislead you," in *Patient Education and Counseling 7(2)*, 1985, pp. 181-191.

[22] Seorld, "Google Update," 05 10 2020. [Online]. Available: https://seorld.com/blog/seo/google-update. [Accessed 21 10 2020].

Targeting Design-Loop Adaptivity

Stephen E. Fancsali
Hao Li
Michael Sandbothe
Steven Ritter
Carnegie Learning, Inc.
{sfancsali, hli, msandbothe,
sritter}@carnegielearning.com

ABSTRACT

Recent work describes methods for systematic, data-driven improvement to instructional content and calls for diverse teams of learning engineers to implement and evaluate such improvements. Focusing on an approach called "design-loop adaptivity," we consider the problem of how developers might use data to target or prioritize particular instructional content for improvement processes when faced with large portfolios of content and limited engineering resources to implement improvements. To do so, we consider two data-driven metrics that may capture different facets of how instructional content is "working." The first is a measure of the extent to which learners struggle to master target skills, and the second is a metric based on the difference in prediction performance between deep learning and more "traditional" approaches to knowledge tracing. This second metric may point learning engineers to workspaces that are, effectively, "too easy." We illustrate aspects of the diversity of learning content and variability in learner performance often represented by large educational datasets. We suggest that "monolithic" treatment of such datasets in prediction tasks and other research endeavors may be missing out on important opportunities to drive improved learning within target systems.

Keywords

Design-loop adaptivity, deep knowledge tracing, Bayesian knowledge tracing, mastery learning, learning engineering.

1. INTRODUCTION

Recent work calls on researchers and developers, including teams of learning engineers [14, 26], to focus on "explanatory" models of learners [25] and "design-loop adaptivity" processes [1, 15] to practically improve learning systems. While researchers describe specific examples of how explanatory learner models and design-loop adaptivity can be used to drive improvements to instruction, less (if any) attention has been paid in the literature to the practical problem of how content developers and learning engineers target and prioritize content for improvement.

We focus on cases in which a target system has a large portfolio of content, elements of which must be prioritized and targeted for improvement given finite learning engineering and software development resources. We present a case study using a data set that is among the largest considered in the literature on knowledge tracing and related methods [9, 18, 22], comprised of middle school and high school student work over an academic year on several hundred mathematics topics, each generally completed by thousands of students, generating several hundred million data points tracking student actions. We motivate, describe, and illustrate two approaches to targeting content for improvement within this portfolio, focusing primarily on what Aleven et al. [1] call "design-loop adaptation to student knowledge," relying on large-scale data to find similarities amongst learners we might leverage to redesign instructional content for better learning.

One targeting method is based on a measure of the extent to which learners tend to struggle with particular pieces of content, and we contrast it with an approach based on the relative prediction performance of deep learning models (i.e., Deep Knowledge Tracing; DKT [18, 22]) compared to traditional Bayesian Knowledge Tracing (BKT; [9]) models.

The first method targets content students struggle to learn, relying on measures of knowledge component (KC [19]; or skill) mastery that are internal to the target intelligent tutoring system (ITS). In contrast, the second method is roughly motivated by the idea that identifying content in which there is a large difference in performance between deep learning and traditional Bayesian approaches may suggest areas in which deep learning can leverage statistical regularities in students' performance that could point to improvements in the KC models that are used to drive adaptation with BKT. Such performance differences may suggest a particular focus area for KC model improvements. Relative DKT performance versus BKT performance also provides an instance of a metric that is perhaps less dependent on how the run-time ITS has "set the bar" for success in terms of KC mastery.

In exploring these two approaches, we illustrate the variability in learning content and experiences within widely deployed systems like Carnegie Learning's MATHia (formerly Cognitive Tutor) [23]. While different facets of variation may at times call for different approaches to content improvement (e.g., variation in student motivation could call for redesigns that discourage "gaming the system" [3]), our present work explores how to guide learning engineers' "attention" to particular pieces of content to then consider specific improvements via processes for design-loop adaptivity [1, 15].

Original contributions of this work are two-fold: (1) We describe a novel problem in the literature related to how to target

Stephen Fancsali, Hao Li, Michael Sandbothe and Steven Ritter "Targeting Design-Loop Adaptivity". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 323-330. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

instructional content improvement or design-loop adaptivity and explore two targeting approaches, and (2) we shed light on opportunities in treating large-scale educational datasets that may be missed by treating such datasets as "monolithic" targets for data-intensive approaches. Treating datasets in a "monolithic" way, though not a universal practice (e.g., [4-5, 10]) may inhibit practical progress in learning engineering.

In addition to considering one of the largest-scale applications of DKT (and BKT) modeling in the literature, we illuminate avenues for research at the intersection of educational data science and learning engineering at scale in a widely-deployed adaptive learning platform for K-12 mathematics. We seek to amplify extant calls for a more nuanced approach to work on performance prediction [15, 25] while illustrating solutions to practical problems in learning engineering and product improvement.

2. DESIGN-LOOP ADAPTIVITY

2.1 Background

A recent survey of adaptive instructional technologies [1] describes three categories along which learners' experience can be varied, including "step-loop adaptivity," "task-loop adaptivity," and "design-loop adaptivity." Step-loop adaptivity and task-loop adaptivity roughly correspond to "inner" and "outer" loop adaptive functionality in ITSs distinguished by VanLehn (e.g., [28**]**), respectively. We briefly describe step-loop and task-loop adaptivity before considering design-look adaptivity.

Step-loop or inner-loop adaptivity enables an adaptive instructional system or ITS to provide support to learners *within a particular learning task* based on their performance (e.g., providing context-sensitive hints or just-in-time feedback within a math problem based on learner responses). Task-loop or outer-loop adaptivity enable an instructional system to *choose the next appropriate task for a learner* based on a model of student learning and evolving estimates of a learner's mastery of underlying competencies, skills, or KCs [19] based on a learner's performance. Extensive educational data mining (EDM) literature considers, for example, variants of and data-driven parameter optimizations for BKT (e.g., [18]), which can be used to select tasks for learners as their mastery of KCs evolves.

In their recent survey, Aleven and colleagues describe design-loop adaptivity as involving

> data-driven decisions made by course designers before and between iterations of system design, in which a… system is updated based on data about student learning, specifically, data collected with the same system… [1].

They go on to describe goals toward which design-loop adaptations might be made, including adaptations to student knowledge, affect and motivation, student strategies and errors, and self-regulated learning, providing examples of each. Canonical examples of design-loop adaptivity or adaptation to student knowledge, the goal of our present targeting and prioritization endeavor, generally involve situations in which content within tutoring systems or online courses are improved by refining the fine-grained KC models that drive the adaptive experience of learners using a combination of data and human expertise [17, 20, 27].

Design-loop adaptivity for motivation and affect might drive content or system design and redesign to discourage off-task behavior [4] and "gaming the system" [3], wherein students attempt to make progress in a system by taking advantage of system features like hints, rather than making genuine attempts to master content. Aleven et al. [1] suggest that an approach to modeling gaming the system behavior based on a large-scale survey of the extent to which gaming the system [3] manifests across topics (what we will refer to as "workspaces") in an intelligent tutoring system like MATHia provides a foundation for future design-loop adaptivity investigations. One important facet of this work (and related work on off-task behavior [4]) is its appreciation of the extent to which there is variability in how learning occurs across different (types of) content within adaptive instructional systems. Appreciating and surveying this variability is vital to ascertaining where, within large portfolios of content, to target design-loop adaptivity efforts and related data-driven, instructional improvement efforts.

2.2 A Process for Design-Loop Adaptivity

Huang et al. [15] describe a systematic approach to design-loop adaptivity or data-driven instructional redesign and improvement. They suggest three general goals for such redesign efforts. For a particular piece of content in an ITS or similar adaptive instructional system with a KC model, the goals are: (1) refine the KC model for the target content, (2) redesign the content, and (3) optimize individualized learning within the content. Existing EDM methods and novel analyses are then described to achieve each of these goals, targeting an "Algebraic Expressions" unit of content within the Mathtutor ITS [2]. For example, KC models can be refined using data-driven, computationally intensive methods like Learning Factors Analysis (LFA; [8]) or a simpler approximation of such an approach that uses regression techniques called "difficulty factor effect analysis" by Huang et al. [15]. Human expertise also plays an important role in such refinements, including in setting up data-driven analyses to produce meaningful results, interpreting these results for inclusion in potential task redesigns, and often in providing suggested refinements for target tasks.

Huang et al. [15] demonstrate that redesigned content improves learning as measured by pre-tests and post-tests. Broadly, these goals align with on-going, data-driven content improvement efforts pursued by learning engineers working with MATHia. Nevertheless, the process of design-loop adaptivity generally requires extensive human and computational resources to be carried out in ways that will drive improved instructional effectiveness. The present work seeks to illustrate how EDM techniques might help improve targeting this process.

3. MATHia

3.1 Learning Platform

Carnegie Learning's MATHia [23] is an ITS used by hundreds of thousands of learners each year, mostly in middle and high school classrooms as a part of a blended math curriculum that combines collaborative work guided by instructors and Carnegie Learning's *MATHbook* worktexts (60% of instructional time in recommended implementations) with individual work in MATHia (40% of instructional time). Nevertheless, usage of MATHia, contexts in which it is used, and other implementation details vary across a diverse, nationwide user-base.

Grade levels of content in MATHia (e.g., Grade 7, Algebra I) are organized into a series of "modules," each of which is comprised of a series of "units." Units are composed of a series of "workspaces." Workspaces represent the underlying unit of learner progress to mastery in MATHia. Each workspace presents

a set of problems associated with a set of KCs; student progress within the system is determined by students' achievement of mastery of all of the KCs associated with a particular workspace, estimated by MATHia using BKT (see §3.2). Learning experiences vary substantially between workspaces with respect to design patterns, content areas, types of practice and instruction provided, (quality of) KC models intended to practice such content (e.g., some the result of years of iterative refinements, others introduced more recently), BKT parameters, and other parameters that drive task selection and mastery judgment.

Consider the problem solving task illustrated in Figures 1 and 2. Figure 1 illustrates the workspace "Modeling the Constant of Proportionality." In this workspace, students are provided with a word problem and several associated questions (left pane; Figure 1). On the right-hand side of Figure 1, tools are presented to solve the problem's "steps." There is a worksheet or table in which they can provide units of measurement, responses to questions, and fields in which to write expressions to model the problem's scenario. After they have completed entries in the worksheet, students work with a graphing tool. Each problem-step in the ITS can provide context-sensitive hints upon request as well as just-in-time feedback that tracks errors that students often make. Most problem-steps are mapped to KCs, for which MATHia provides an evolving mastery estimate to adapt problem selection to the individual student's needs (see §3.2).

Contrast the learning experience of the problem in Figure 1 with that of Figure 2. "Modeling the Constant of Proportionality" (Figure 1) involves substantive reading, modeling the problem scenario via algebraic expressions, working through concrete instances of these expressions, and using a graphing tool. Figure 2 illustrates problem-solving in a menu-based equation "solver" workspace, "Solving with the Distributive Property Over Multiplication." Here the student is tasked with solving for x in the equation $65 = 10\,(x + 6)$. There is little reading and no context provided for the equation, but hints and just-in-time feedback are available. Learners' progress toward mastery is tracked for a different set of KCs. The menu-based solver constrains possible student actions at various points in the equation-solving process compared to the typed-in input that students provide in the worksheet in Figure 1. Far from an exhaustive list, we seek to illustrate a few from among substantial differences in types of content provided, design patterns, interaction modalities, underlying KC models, and tools available, even within the relatively constrained domain of math, any of which may have important impacts on inferences that might be drawn from data or the ability of different methods to predict performance and learning within such content. While any of the features in these examples might reasonably be refined as a part of the design-loop adaptivity or content improvement process, we leave to future work the data-driven targeting of specific improvements within a workspace. We consider how to target specific "workspaces" for design-loop adaptivity improvements.

3.2 Knowledge Tracing & Mastery Learning

BKT [9] posits a binary (i.e., "mastered" or "unmastered") knowledge state for each independently modeled KC and can be formalized as a four-parameter hidden Markov model. One parameter represents the probability that a learner has already mastered a KC before their first opportunity to practice it. A second parameter represents the probability that a learner transitions from the unmastered to the mastered state at any particular KC practice opportunity. Two parameters link the knowledge state to observable outcomes at any KC practice opportunity: the probability that a student is in the unmastered state and responds correctly ("guessing") and the probability that a student is in the mastered state and answers incorrectly ("slipping"). Extensive EDM literature has explored the data-driven fitting of BKT parameters as well as individualized (e.g., [30]) and more sophisticated variants of this approach (e.g., [18]).

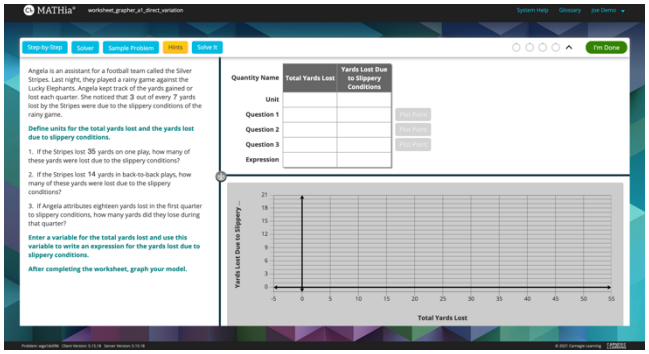


Figure 1. Problem-solving screenshot from a MATHia workspace called "Modeling the Constant of Proportionality."

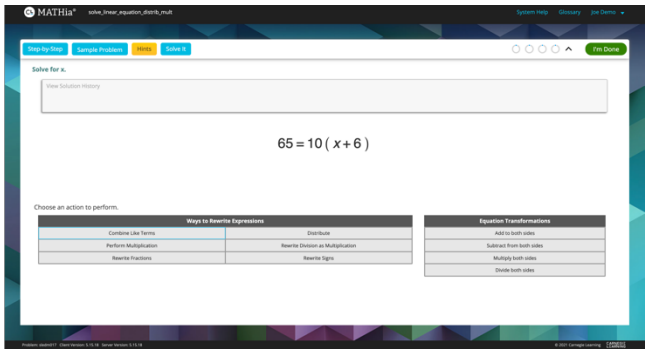


Figure 2. Screenshot from the MATHia workspace "Solving with the Distributive Property Over Multiplication."

Based on parameter settings and performance data collected as a student practices each KC, the system can use BKT to infer and update estimates of the probability that a student is in the "mastered" state for any particular KC. Typically, systems set a threshold for mastery (often 0.95, as in MATHia); if the system's estimate that the probability a student has mastered a particular KC is above the threshold, then the system considers the KC mastered for that student.

Relying on evolving estimates of learner KC mastery, instructional systems can use knowledge tracing frameworks like BKT to drive "task-loop" (or "outer loop") adaptivity [1, 28] and mastery learning [7, 24]. After a student completes a problem (or task; like the problems illustrated in Figures 1-2), the system can select the next problem based on KCs that a student has yet to master. In this way, systems can adapt to the student's evolving mastery of KCs, providing (ideally) just enough practice for students to master KCs and avoiding cases in which the system provides too little or too much practice.

Implementing self-paced mastery learning [7, 24], MATHia provides practice to a student until they have either mastered all KCs associated with a particular workspace or they have reached the maximum number of problems that designers have specified for a particular workspace. Once the student masters all of the KCs in a particular workspace (or reaches the max number of problems), they are moved on to the next workspace in an

assigned content sequence. Teachers are alerted when students reach the max number of problems in a workspace without reaching mastery. Setting a max number of problems ensures that students do not endlessly struggle unproductively within a piece of content [11].

3.3 Data

We consider data from 252,036 learners who used MATHia during the 2018-19 academic year and completed at least one of 308 workspaces that track KC mastery across math content for Grades 6-8, Algebra I, Algebra II, and Geometry. These data account for approximately 3.8 million workspace completions. Models are learned over subsets of 267,419,999 student actions (i.e., first-attempts, including hint requests) at problem-steps mapped to KCs. Over the 308 workspaces, MATHia tracks 2,152 KCs. Table 1 provides summary statistics.

Table 1. Summary statistics for 308 MATHia workspaces in 2018-2019; "KCs" = # KCs tracked; "Comps." = # student-workspace-completions; "Actions" = sum across all students completing workspace of count of first attempts (including hint requests) at problem-steps within workspace problems.

	Min.	Q1	Med.	Q3	Max.
KCs	2	5	6	9	15
Comps.	167	4275	9414	18801	51097
Actions	5530	197757	489159	1278325	7191034

When working with large, complex datasets, it is essential to focus learning engineering efforts on the portions of the system for which improvements can be most impactful. Rather than consider such a broad dataset as a single monolithic target, especially for performance prediction modeling in §4.2, we learn models for each workspace within the dataset; input data are sequences of correctness labels for learner actions (e.g., binary correct or incorrect, where incorrect includes both errors and hint requests) and labels for KCs mapped to each action.

4. METRICS FOR TARGETING IMPROVEMENTS

As illustrated in Figures 1 and 2, workspace-to-workspace variability in learning experiences is substantial. Types of practice vary (e.g., equation solving, graphing, etc.), and developers make a plethora of design choices in creating content. Some workspaces require more reading; KC models vary in complexity, and some have been iteratively refined over the course of nearly two decades while others are newly deployed in a given year. Given this variation and the nature of grade-level content standards, there is also variability in the extent to which learners find particular content difficult.

Learner difficulties manifest at the problem-step level in the form of problem-solving errors and hint requests and at the workspace level in at least two ways: (1) that some learners require a greater number of problems to achieve mastery of all KCs, and (2) that some learners reach the maximum number of problems set by designers without having achieved mastery of all KCs. These latter students are moved along within their curriculum sequence without mastery. Teachers are alerted of this failure to reach mastery via reporting analytics available to them as well as in the LiveLab teacher companion app to MATHia. Some students fail to reach mastery in a workspace because of genuine difficulty with presented math content, but relatively frequent instances of such failure to reach mastery often indicate that content improvements (i.e., design-loop adaptivity) is called for to enhance experiences for learners.

Prior research considers MATHia's workspace level as a unit of analysis. Researchers have focused on associations between characteristics of Cognitive Tutor "lessons" (MATHia's workspaces) and learners' affective states like confusion and frustration [10] as well as the extent to which students go off-task [4] and game the system [5]. In what follows, we adopt an approach similar in spirit to this literature by considering a large corpus of MATHia data as broken down into workspaces rather than treating the entire dataset in a monolithic fashion.

The first metric we consider helps identify content that is instructionally ineffective in ways that manifest as difficulty for learners to successfully complete the content. In considering the second metric, we explore one example where the metric may be providing some insights into places where content is not "difficult" (i.e., measures of difficulty do not "raise flags" about improvement needs) but where design-loop adaptivity improvements might drastically improve student learning.

4.1 Proportion of Failures to Reach Mastery

The first design-loop adaptivity targeting metric we consider is the proportion of learners who fail to reach mastery of at least one of the KCs associated with a workspace before reaching the maximum number of problems set by content designers. Figure 3 provides a histogram showing the overall distribution of this proportion across workspaces. The median workspace has 4.3% of students fail to reach mastery of all its KCs (minimum = 0%; Q1 = .7%; Q3 = 12.1%; maximum = 77.7%).

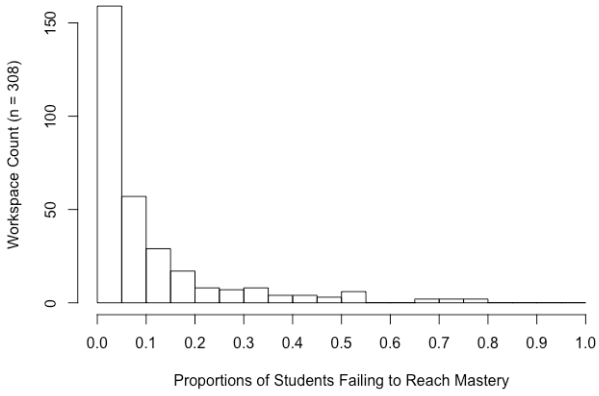


Figure 3. Histogram illustrating the distribution of the proportion of students failing to reach mastery of all KCs associated with 308 workspaces in the 2018-19 academic year.

Fancsali et al. [11] argue that students' failure to achieve mastery at a level of aggregation like that of a workspace is an important outcome for predictive modeling, mostly overlooked in the literature on so-called "wheel spinning" (e.g., [6]), which tends to develop models to predict whether students will master particular KCs in a tutoring system, ignoring other elements of how instructional content is presented. Fancsali et al. argue that, given the clustering of KCs within problems, the clustering of problems within workspaces, and the fact that workspaces are the unit at which learners make progress in ITSs like MATHia, reporting outcomes like the count and percentage of KCs that student fail to master (a la Beck and Gong [6]) is of dubious practical value.

Since design-loop adaptivity improvements are likely to often involve redesign of instructional content, we similarly contend that measures closely aligned to instructional delivery are likely to be helpful in targeting this process. Large proportions of students failing to master instructional content are likely to be important in determining what learning content to improve with limited resources. This metric serves as a foil to a second approach.

4.2 DKT vs. BKT Prediction Performance

Extensive recent literature (e.g., [12, 18, 22]) considers deep learning approaches to the problem of predicting student performance at fine-grained opportunities to demonstrate mastery of KCs in learning systems like ASSISTments [13]. DKT [22] has been compared (e.g., [12, 18] to BKT and logistic regression approaches to the same type of prediction task [21]. Early work demonstrated that DKT generally had superior prediction performance compared to BKT [22], but subsequent literature also suggests that variations of BKT (e.g., modeling "forgetting") and logistic regression approaches can bridge some, if not most, of the gap in prediction performance (e.g., [18, 29]).

Nevertheless, we seek to better understand the extent to which DKT out-performs BKT when considered workspace-by-workspace across a large dataset from MATHia, which presents a wide variety of learning experiences. We find that, for a variety of workspaces, classic BKT's performance is often comparable to DKT even without accoutrements added in the work of Khajah et al. [18]. Further, in keeping with our primary concern in the present work, we explore the extent to which observed differences in performance between the two approaches, especially examples of DKT's far superior prediction performance, might serve as a metric for targeting improvement work for MATHia workspaces, possibly indicating an especially flawed KC model.

4.2.1 Modeling Approach

We rely on the Khajah et al. [18] implementation of DKT with long short-term memory (LSTM) recurrent units.[1] We use Yudelson's *hmm-scalable*[2] implementation of classic BKT parameter fitting using expectation maximization [30]. We learn DKT and BKT models for each of the 308 workspaces, splitting the data for each workspace into training and test sets with a 80%-20% student-level split and calculate the AUC (area under the receiver-operating characteristic curve) on the test set following methods in Khajah et al. [18]. BKT and DKT models are trained and tested on the same datasets. AUC is a measure of the extent to which a model can "discriminate" between or predict students' correct and incorrect responses in the held-out test set. An AUC value of 0.5 indicates "chance" ability to discriminate between two classes; a value of 1.0 indicates perfect discrimination.

4.2.2 Results

Table 2 provides summary statistics for AUC performance for DKT, BKT, and AUC differences of these methods over all workspaces. As expected, DKT generally provides superior prediction performance to classic BKT over the 308 workspaces. However, there is substantial variability, with classic BKT in some cases, albeit many (but not all) with relatively small sample sizes, even out-performing DKT. While there is a modest, statistically significant positive correlation between the AUC difference in DKT and BKT and sample size (i.e., the number of student-sequences available for training and testing) ($r = .2$; $p < .001$), BKT performs comparably to DKT on a number of workspaces with tens of thousands of students' data, and BKT only underperforms DKT by approximately .07 AUC units for the median workspace. The Q1 value for this difference (the greatest difference over 77 workspaces) is approximately in line, in terms of AUC units, with a value (.03 AUC units) declared comparable by Khajah et al. [18] for BKT "variants" compared to DKT.

The difference in AUC between DKT and BKT is uncorrelated with the proportion of students who fail to reach mastery ($r = -.05$; $p = .4$) and is thus not an indicator of the relative difficulty of particular workspaces, regardless of the source of difficulty.

Table 2. Summary statistics for AUC performance over 308 workspaces of DKT and BKT models and of the difference between DKT and BKT performance (Δ); negative minimum value indicates better BKT performance for some workspaces.

AUC	Min.	Q1	Med.	Q3	Max.
DKT	.5852	.7839	.8331	.8783	.9763
BKT	.5150	.7045	.7456	.7854	.9563
Δ	-.0802	.0361	.0676	.1281	.3073

4.2.3 Practical Promise

We consider two observations relating to workspace design patterns that emerge from considering workspaces with the largest differences in terms of DKT's (generally better) prediction performance compared to BKT. First, we consider the design of a particular workspace as a prime target for design-loop adaptivity to student knowledge, motivation, and affect. Second, we consider more general design patterns in workspaces on which DKT and BKT performance differences are greatest, suggesting more "macro-level" design-loop adaptivity that may affect broader categories of workspaces.

4.2.3.1 Example Workspace

The second greatest observed difference in AUC occurred for the workspace "Checking Solutions to Linear Equations" (DKT AUC = .968; BKT AUC = .684). A mere 0.2% of students fail to master all KCs in this workspace, suggesting that it may not be "flagged" for design-loop adaptivity improvements based on difficulty. Nevertheless, careful inspection of the workspace yields several areas for improvement.

This workspace presents students with problems (See Figure 4) like: "Jordan solved the equation $-3u - 8 = 10$. She calculated $u = -6$. Use the Solver to check Jordan's solution." The student is then presented with a menu-based equation solver. Work with the equation solver should involve the student substituting in the solution value from the problem presentation and checking whether the result is a balanced equation. After choosing "Substitute for variable" from the menu, the student then must input a value on the left-hand side of the equation (see Figure 5).

Problems in this workspace present both correct and incorrect cases, but the KC model does not distinguish between correct and incorrect cases, making problems with a correct solution targets for possible gaming the system. For example, in the problem in Figure 5, the student might enter *10* to complete "*10 = 10*." This response may not reflect having correctly carried out the variable substitution to arrive at this solution. KCs in this workspace are also not currently mapped to work in the solver; the solver provides hints and just-in-time feedback on errors, but it is not instrumented to track KC mastery. Once the student has entered

[1] https://github.com/mmkhajah/dkt

[2] https://github.com/myudelson/hmm-scalable

the appropriate value, two questions appear on the left of the screen (see Figure 6), and student responses to these questions trigger updates to two KCs at a time.

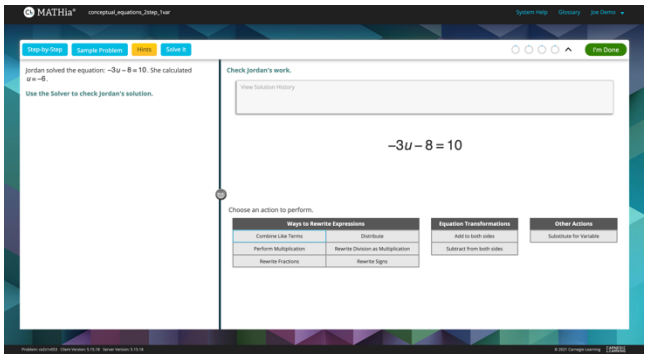


Figure 4. Screenshot in the MATHia workspace "Checking Solutions to Linear Equations."

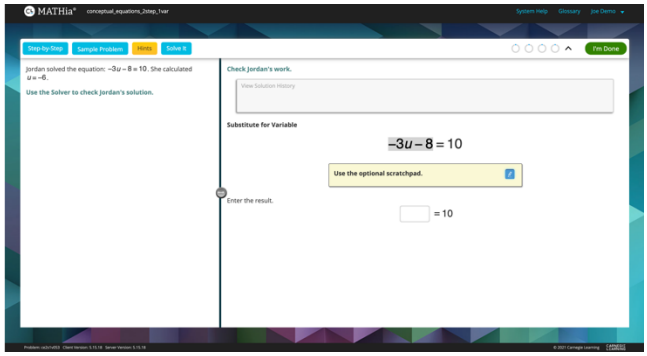


Figure 5. Screenshot after the student has selected "Substitute for variable" from the equation solving menu (see Figure 4).

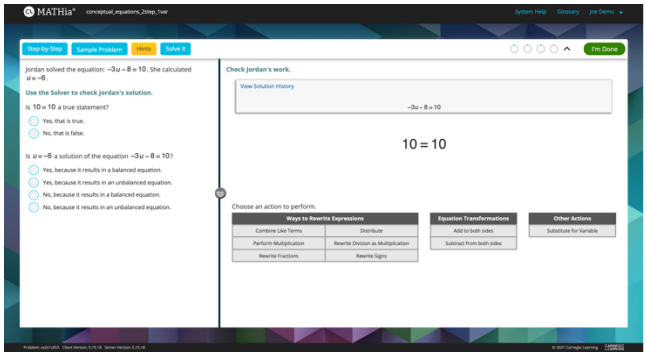


Figure 6. Screenshot after the student has entered the value 10 on the left-hand side of the equation (see Figure 5).

Following design-loop adaptivity steps laid out by Huang et al. [15], we have identified (1) where the KC model can be refined, and (2) areas for task redesign. The third step would involve fitting a BKT model using the hypothetical re-mapping of KCs to steps within problems in existing data to determine whether the hypothesized, refined KC model fits the data better than the existing KC model. Future work will experimentally test workspace redesigns to "close the loop" (cf. [16, 25]) between this data-driven approach and empirical learning outcomes.

4.2.3.2 Prominent Design Patterns

Patterns emerge in comparing the performance of DKT to BKT over 308 workspaces. In the top twenty workspaces in which DKT outperforms BKT, differences in AUC units range from .307 to .212, and all provide constrained input mechanisms relative to broader MATHia content. Fourteen workspaces (70%) involve equation solving, and the others are split between those that involve placing values on a number line and those in which problem input is provided via drop-down menus.

Gervet et al. raise questions about explanations for observed properties of DKT in predicting student performance. Can DKT, for example, "better pick up on local patterns of student behavior like gaming the systems" [12]? While far from conclusive, DKT's performance for the workspace "Checking Solutions to Linear Equations" could exemplify this phenomenon. Workspaces with more constrained inputs may provide examples where DKT "picks up" on local patterns that BKT does not. Future work ought to investigate whether these particular types of relatively constrained input mechanisms are easy to "game" or whether and how DKT learns local performance patterns.

Equation solver and number line workspaces are widespread in the top workspaces in which DKT outperforms BKT. "Checking Solutions to Linear Equations" has readily apparent flaws, suggesting that our approach may be promising in targeting instructional improvement work. Systematic review of these results remains future work.

5. DISCUSSION

There are numerous questions for future research. That differences in AUC between DKT and BKT are uncorrelated with an important measure of instructional ineffectiveness, combined with DKT's ability to find regularities in data that are not found by BKT suggests that this difference may be signaling important workspace characteristics. Analysis of a particular workspace (§4.2.3.1) suggests that DKT-BKT differences may signal inadequacies in the KC model. These findings can be compared to the results of data-driven search for better KC models [8]. Improvements can be made to the workspace, and A/B tests can "close the loop" and establish more effective approaches.

Systematic analysis of instructional content and prediction performance differences in DKT and BKT might follow work that explores a space of properties and features of particular tutor "lessons" to determine which predict students' affect, gaming the system, and off-task behavior [4-5, 10]. Comparisons to logistic regression methods (e.g., [12]) are also needed.

Naïve learning engineering may focus on reducing students' mastery failures. Such an approach could lead to "over-simplified" tasks that don't produce failure because they don't require much knowledge. Large differences between DKT and BKT may help identify over-simplified workspaces that provide opportunities for students to game the system [3, 12]. To what extent do gaps in modeling techniques' performance indicate unproductive patterns of "local" behavior in particular workspaces? What else drives differences? What other behavior patterns indicate ways to target improvement?

Methodologically, our "non-monolithic" analysis of a large educational data set treats component instructional experiences as units for analysis. Such analytical decomposition is vital to practical learning engineering to improve instructional systems and large portfolios of content used by learners every day.

6. ACKNOWLEDGMENTS

This research was sponsored by the National Science Foundation under the award The Learner Data Institute (Award #1934745). The opinions, findings, and results are solely those of the authors and do not reflect those of the National Science Foundation.

7. REFERENCES

[1] Aleven, V., McLaughlin, E.A., Glenn, R.A., Koedinger, K.R. 2017. Instruction based on adaptive learning technologies. In *Handbook of Research on Learning and Instruction*, 2nd Ed., Routledge, New York, 522–560.

[2] Aleven, V., Sewall, J. 2016. The frequency of tutor behaviors: a case study. In *ITS 2016*. LNCS, vol. 9684, Springer, Cham, 396–401. https://doi.org/10.1007/978-3-319-39583-8_47

[3] Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. 2004. Off-task behavior in the Cognitive Tutor classroom: When students "game the system." In *Proceedings of ACM CHI 2004: Computer-Human Interaction*, 383-390.

[4] Baker, R.S.J.d. 2009. Differences between intelligent tutor lessons, and the choice to go off-task. In *Proceedings of the 2nd International Conference on Educational Data Mining*, 11-20.

[5] Baker, R.S.J.d., de Carvalho, A.M.J.A., Raspat, J., Aleven, V., Corbett, A.T., Koedinger, K.R. 2009. Educational software features that encourage and discourage "gaming the system". In *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, 475-482.

[6] Beck J.E., Gong Y. 2013. Wheel-Spinning: Students Who Fail to Master a Skill. In *Artificial Intelligence in Education.* AIED 2013. LNCS, vol 7926. Springer, Berlin/Heidelberg. https://doi.org/10.1007/978-3-642-39112-5_44

[7] Bloom, B. S. 1968. Learning for mastery. *Evaluation Comment* 1(2). Los Angeles: University of California at Los Angeles, Center for the Study of Evaluation of Instructional Programs.

[8] Cen, H., Koedinger, K.R., Junker, B. 2006. Learning factors analysis: A general method for cognitive model evaluation and improvement. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* Springer-Verlag, Berlin, 164–175.

[9] Corbett, A.T., Anderson, J.R. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction* 4, 253– 278.

[10] Doddannara L.S., Gowda S.M., Baker R.S.J.., Gowda S.M., de Carvalho A.M.J.B. 2013. Exploring the relationships between design, students' affective states, and disengaged behaviors within an ITS. In *Artificial Intelligence in Education.* AIED 2013. LNCS, vol 7926. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-39112-5_4

[11] Fancsali S.E., Holstein K., Sandbothe M., Ritter S., McLaren B.M., Aleven V. 2020. Towards practical detection of unproductive struggle. In *Artificial Intelligence in Education*. AIED 2020. LNCS, vol 12164. Springer, Cham. https://doi.org/10.1007/978-3-030-52240-7_17

[12] Gervet, T., Koedinger, K., Schneider, J., & Mitchell, T. (2020). When is deep learning the best approach to knowledge tracing? *Journal of Educational Data Mining* 12(3), 31-54. https://doi.org/10.5281/zenodo.4143614

[13] Heffernan, N.T., Heffernan, C.L. 2014.The ASSISTments ecosystem: building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *Int. J. Artif. Intell. Educ.* 24(4), 470–497.

[14] Hess, F., Saxberg, B. 2014. *Breakthrough leadership in the digital age: Using learning science to reboot schooling.* Thousand Oaks, CA: Corwin.

[15] Huang Y., Aleven V., McLaughlin E., Koedinger K. 2020. A general multi-method approach to design-loop adaptivity in intelligent tutoring systems. In *Artificial Intelligence in Education.* AIED 2020. LNCS, vol 12164. Springer, Cham, 124-129. https://doi.org/10.1007/978-3-030-52240-7_23

[16] Liu, R., & Koedinger, K.R. (2017). Closing the loop: Automated data-driven cognitive model discoveries lead to improved instruction and learning gains. *Journal of Educational Data Mining* 9(1), 25–41.

[17] Lovett, M., Meyer, O., Thille, C. 2008. The open learning initiative: Measuring the effectiveness of the OLI statistics course in accelerating student learning. *Journal of Interactive Media Education*, 14.

[18] Khajah, M., Lindsey, R.V., Mozer, M.C. 2016. How deep is knowledge tracing? In *Proceedings of the 9th International Conference on Educational Data Mining (Jun 29 - Jul 2, 2016, Raleigh, NC, USA)*. EDM 2016. International Educational Data Mining Society, 94-101.

[19] Koedinger, K.R., Corbett, A.T., Perfetti, C. 2012. The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science* 36, 5, 757–798.

[20] Koedinger, K.R. & McLaughlin, E.A. 2010. Seeing language learning inside the math: Cognitive analysis yields transfer. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*. Austin, TX, Cognitive Science Society, 471-476.

[21] Pavlik, P.I., Cen, H., Koedinger, K.R. 2009. Performance factors analysis – a new alternative to knowledge tracing. In *Proceedings of the 2009 Conference on Artificial Intelligence in Education*. IOS Press, 531–538.

[22] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, 505–513.

[23] Ritter, S., Anderson, J.R., Koedinger, K.R., and Corbett, A.T. 2007. Cognitive Tutor: applied research in mathematics education. *Psychon. B. Rev.* 14, 249-255.

[24] Ritter, S., Yudelson, M., Fancsali, S.E., Berman, S.R. 2016. How mastery learning works at scale. In *Proceedings of the 3rd Annual ACM Conference on Learning at Scale* (Apr 25 - 26, 2016, Edinburgh, UK). L@S 2016. ACM, New York, NY, 71-79.

[25] Rosé, C.P., McLaughlin, E.A., Liu, R., Koedinger, K.R. 2019. Explanatory learner models: Why machine learning (alone) is not the answer. *Br J Educ Technol* 50, 2943-2958. https://doi.org/10.1111/bjet.12858

[26] Simon, H.A. 1967. The job of a college president. *Educational Record* 48(Winter), 68–78.

[27] Stamper, J. & Koedinger, K.R. 2011. Human-machine student model discovery and improvement using data. In *Proceedings of the 15th International Conference on Artificial Intelligence in Education*. Springer, Berlin/Heidelberg, 353-360.

[28] Vanlehn, K. 2006. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227–265.

[29] Wilson, K.H., Xiong, X., Khajah, M., Lindsey, R.V., Zhao, S., Karklin, Y., Van Inwegen, E.G., Han, B., Ekanadham, C., Beck, J.E., Heffernan, N., Mozer, M.C. 2016. Estimating student proficiency: Deep learning is not the panacea. In *Proceedings of the Workshop on Machine Learning for Education at the 30th Conference on Neural Information Processing Systems (NIPS 2016).*

[30] Yudelson, M., Koedinger, K., Gordon, G. 2013. Individualized Bayesian Knowledge Tracing Models. In *Proceedings of 16th International Conference on Artificial Intelligence in Education* (Memphis, TN) AIED 2013. LNCS vol. 7926, Springer-Verlag, Berlin/Heidelberg, 171-180.

Going Online: A simulated student approach for evaluating knowledge tracing in the context of mastery learning

Qiao Zhang
Drexel University
Philadelphia, PA, USA
qiao.zhang@drexel.edu

Christopher J. Maclellan
Drexel University
Philadelphia, PA, USA
christopher.maclellan@drexel.edu

ABSTRACT

Knowledge tracing algorithms are embedded in Intelligent Tutoring Systems (ITS) to keep track of students' learning process. While knowledge tracing models have been extensively studied in offline settings, very little work has explored their use in online settings. This is primarily because conducting experiments to evaluate and select knowledge tracing models in classroom settings is expensive. To fill this gap, we introduce a novel way of using machine-learning models to generate simulated students. We conduct experiments using agents generated by the Apprentice Learner Architecture to investigate the online use of different knowledge tracing models (Bayesian Knowledge Tracing, the Streak model, and Deep Knowledge Tracing). An analysis of our simulation results revealed an error in the initial implementation of our Bayesian knowledge tracing model that was not identified in our previous work. Our simulations also revealed a more fundamental limitation of Deep Knowledge Tracing that prevents the model from supporting mastery learning on multi-step problems. Together, these two findings suggest that Apprentice agents provide a practical means of evaluating knowledge tracing models prior to more costly classroom testing. Lastly, our analysis identifies a positive correlation between the Bayesian knowledge tracing parameters estimated from human data and the parameters estimated from simulated learners. This suggests that model parameters might be initialized using simulated data when no human-student data is yet available.

Keywords

Computational Models of Learning, Simulated Students, Knowledge Tracing

1. INTRODUCTION

Intelligent Tutoring Systems (ITS) are used within K-12 education to improve learning outcomes. In addition to providing students with scaffolding and feedback, tutors utilize an approach called *knowledge tracing* to estimate what students know and do not know [2]. When combined with a problem selection policy [16], knowledge tracing enable tutors to support mastery learning and to focus students practice where it is most needed (i.e., on the skills they do not yet know rather than the skills they already know). While many studies have explored knowledge tracing for offline evaluation (fitting knowledge tracing models to existing data sets), there is comparatively little work on evaluating these algorithms in online settings (evaluating how well these algorithms estimate students' mastery from just a few data points and decide when to stop giving them additional problems).

We aim to understand which knowledge tracing models yield the greatest mastery learning efficiency in online settings. Additionally, we want to find out how the parameters for knowledge tracing models can be selected before human data is collected. To meet our need for multiple experiments to investigate our knowledge tracing questions, we introduce a novel way of using computational models of learning, or simulated student models that learn from interactions with a tutor just like human students do, to simulate our knowledge tracing experiments. We use the Apprentice Learner architecture [10], a machine-learning framework that aims to model how humans learn from examples and feedback to generate simulated students and conduct experiments.

To explore the feasibility of this approach, we conducted experiments to compare Bayesian Knowledge Tracing (BKT)[2] to the Streak model [7] and Deep Knowledge Tracing (DKT) [15]. Our simulations show that both BKT and Streak stop before giving all the problems, but that BKT is slightly more aggressive than Streak and seems to assume students have mastered skills a bit earlier than expected. Upon further inspection, our analysis revealed a bug in our underlying implementation of BKT (which we fixed for this study). Further, we found that DKT exhibits strange behavior that makes it unusable in certain cases of mastery learning and problem selection. This limitation of DKT for mastery learning has not been identified in prior work. These findings demonstrate that simulation students might serve a valuable role in testing knowledge tracing models before more costly classroom deployments.

We also explore the use of simulated data from these experiments to estimate initial parameters for the BKT model. Prior to collecting human data, knowledge tracing parameters are often set to reasonable hand-picked defaults. A better approach is to run a pilot study with human students

Qiao Zhang and Christopher MacLellan "Going Online: A simulated student approach for evaluating knowledge tracing in the context of mastery learning". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 331-337. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

to collect data for model training, which requires additional time and labor. Our analysis identifies a positive correlation between the BKT parameters estimated from the simulated data and those estimated from human data, suggesting simulated data might be used to initialize parameters.

2. BACKGROUND

2.1 Knowledge Tracing

The main purpose of knowledge tracing is to track students' skill mastery and predict their future performance based on their past activity. Knowledge tracing uses labels of the skills needed for each step, which we call Knowledge Components (KCs), and students' first attempt correctness (correct or incorrect) to predict whether the students will correctly solve a new problem containing the same KC.

2.2 Bayesian Knowledge Tracing

BKT [2] is a well-known knowledge tracing algorithm that can estimates whether students have learned a particular skill given their past performance on opportunities to practice that skill. It models student knowledge using a Hidden Markov Model, where the hidden state is estimated from observations of students' correctness on each step. For each skill, a student can be in one of the two possible knowledge states: "unknown" or "known". A binary response (correct or incorrect) is generated at each opportunity a student practices a skill [5]. Although, BKT supports the ability to model forgetting, it is typically assumed that students never forget what they have mastered [16]. If a student reaches 95% probability of being in the known state for a skill, the skill is marked as being mastered by the student. With these assumptions, the BKT model has four parameters.

- P(L_0): initial probability of mastery ("known").
- P(T): probability of learning the skill ("learn").
- P(G): probability of guessing the answer ("guess").
- P(S): probability of making a mistake ("slip").

Researches have created variants of the BKT model. Yudelson et al. [20] introduced an individualized BKT model that can take student differences in initial mastery and skill learning probabilities into account. Nedungadi et al. [13] created PC-BKT (Personalized and Clustered), which has individual priors for each student and skill, and dynamically clusters students based on learning ability. This prior work aims to improve predictive performance over original BKT. Despite the quantity of research on BKT models, there is relatively little evaluation in online settings.

2.3 Streak Model

Another knowledge tracing approach that is popular for use within mastery learning is the Streak model. Also known as "three-in-a-row" [7], it is a relatively simple and intuitive model since it only has one parameter, how many correct answers in a row equates to mastery. It was first applied in ASSISTments and the key idea was to keep giving the student questions until some proficiency threshold was reached. The default setting was "three correct in a row" but this could be manipulated by teachers.

2.4 Deep Knowledge Tracing

DKT [15] is a knowledge tracing model that has been receiving increasing attention. This model leverages information about students sequence of steps and correctness on those steps to predict performance on subsequent steps. Additionally, DKT leverages information about performance on one skill to improve predictive performance on other skills. DKT uses a long short-term memory (LSTM) architecture, which is a kind of recurrent neural network that allows for the modeling of non-Markovian processes. Recent work evaluating DKT [6, 8] suggests that DKT often outperforms BKT in terms of predictive performance. Despite this promising finding, there has been very little work exploring the use of DKT within online knowledge tracing (e.g., for mastery learning within a tutor). Beyond the original DKT work [15], which explores the use of DKT for next step recommendation, we are unaware of any research programs that currently uses DKT in this way.

Finally, it is worth noting that knowledge tracing is not strictly necessary for tutoring systems. Many tutors either use a fixed problem sequence or present a fixed number of problems in a random order. However, we hypothesize that knowledge tracing in conjunction with mastery learning component is one of the main components of tutors that makes tutors effective.

2.5 Computational Model of Learning

The Apprentice Learner Architecture is a framework for modeling human learning from demonstrations and feedback in educational environments [10]. We use an Apprentice model previously developed in prior work; see [11] for a complete description of the model. Most work in the field of educational data mining focuses on building mathematical, predictive models of learning. In contrast, the Apprentice models actually perform the task (not just predict performance). They induce task-specific knowledge from the demonstrations and feedback they receive. Apprentice models are ideal for the current study because they do not require prior human data to operate. They can predict learning and behavior based solely on the task structure.

3. METHODOLOGY

We created 30 simulated students (Apprentice agents) to solve problems in a fraction arithmetic tutor (tutor presented in [14]). The tutor had three different types of problems: Add Different (AD), add fractions with different denominators; Add Same (AS), add fractions with same denominators; Multiplication (M), multiply two fractions.

3.1 Experiment Design

Our study had six conditions: Random, Streak, BKT_default, BKT_random, BKT_human and DKT_random. There were four types of conditions: Random, BKT, Streak, and DKT, which differ in the way they select the next problem to give to a simulated student. In the Random condition every problem was assigned only once in random order and the training ends when problems run out. Since Random gives the most training, it produces the highest correctness prediction by the end of practice. We use Random as a baseline for evaluating other models.

The other conditions use the respective knowledge tracing approaches for mastery learning and problem selection. During problem selection, each knowledge tracing model randomly chooses a problem with at least one unmastered skill

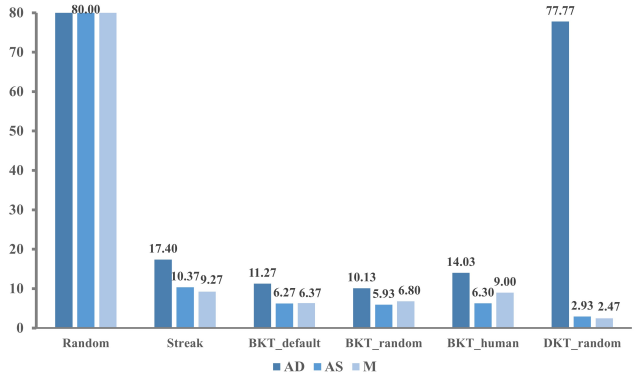


Figure 1: Num. problems given in each condition (by type).

and updates the student's mastery level based on the result. The training ends once the proficiency threshold is reached (95% in all cases but Streak, where it was 3 in a row).

The parameters in BKT_default were manually set based on our prior experience with BKT. We set P(L_0) to 0.1, P(T) to 0.05, P(G) to 0.05 and P(S) to 0.02. These parameters are identical for each KC. The BKT_random and BKT_human parameters were estimated using the BKT module on LearnSphere [9]. The human-based parameters were obtained from fitting BKT to the "Fraction Addition and Multiplication" dataset accessed via DataShop (Koedinger et al., 2010). The random-based parameters were obtained from fitting BKT to the log data generated from simulated students in the Random condition.

To support the use of DKT within online master learning, we created our own implementation using PyTorch's LSTM module.[1] Based on prior work [8], the model has 200 nodes in the hidden layer, uses a dropout of 0.4 during training, and uses a batch size of 5 (our sequences were longer than those in prior work, so a smaller batch size works well). This implementation supports the ability to fit DKT to data presented in standard DataShop [9] format. Trained models have a simple interface for use in online knowledge tracing settings. Similar to BKT, we fit DKT to the log data generated from simulated students in the Random condition to estimate model parameters.

3.2 Simulation Studies and Evaluation

During the experiment process, we created 30 simulated students for each of the six conditions and analyzed the data that they generated. For these experiments, we created a KC model that labels each step as a combination of "Problem Type" and "Selection". There are 14 unique KCs in our analysis, 8 for Add Different, 3 for Add Same and 3 for Multiplication. As the Additive Factors Model (AFM) is often used to examine learning curves from existing data [1], we used pyAFM [12] (a python implementation of AFM) to predict the probability that students will get a next step with the respective skill correct at the end of their practice. This provided an independent means for us to estimate how well each knowledge tracing approach did at appropriately recognizing when students had achieved mastery.

[1]Open-source code for the model is available here: https://gitlab.cci.drexel.edu/teachable-ai-lab/dkt_torch.

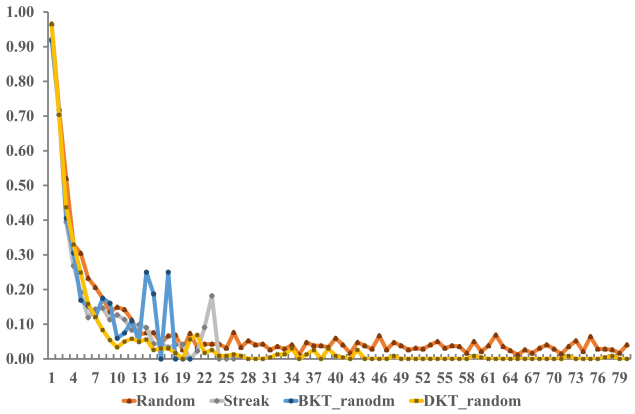


Figure 2: Learning curves for four conditions.

The AFM model assumes that performance monotonically converges to zero error in the tail. However, both humans and simulated students have non-zero error in the tail of their learning curve. This violation of the AFM model's assumption causes the model to estimate lower learning rates in order to accommodate non-zero error in the tail. We found that because the simulated students sometimes get a much larger number of practice opportunities than human students (e.g., 80 vs. 30 practice opportunities), the bias in AFM's learning rates was non-trivial. To address this challenge, we utilized the Additive Factors Model + Slip (AFM+S) approach [12], which explicitly models non-zero error rates in the tail using additional "slipping" parameters for each KC. The AFM+S model better fit the simulated student data from all six conditions than the AFM model (three fold cross-validated RMSE = 0.240 vs. 0.257). Qualitatively, we found that the AFM+ S learning curves seemed to better fit the data, particularly for slopes at the beginning of difficult to master skills.

4. SIMULATION RESULTS

4.1 Online Knowledge Tracing Results

Figure 1 shows the numbers of problems administered by the tutoring system in each condition. Random always gives all 80 problems each type. Streak gives around 17 problems for AD, 10 problems for AS and 9 problems for M. BKT_default gives around 11 problems for AD and 6 problems for AS and M. BKT_random has the similar statistics to the BKT_default, while BKT_human gives around 14 AD problems, 9 M problems and around 6 AS problems. DKT_random gives around 78 AD problems, almost as many as Random; however, it gives less than 3 problems in AS and M. The number of problems given by BKT_human is slightly higher than those given by BKT_random. We hypothesize that this is because the BKT_random parameters were fit specifically to the simulated students, so when used for knowledge tracing they provide better estimates of mastery than the the BKT_human parameters.

To get a better sense of the overall differences between Random, Streak, BKT (BKT_random), and DKT, we plotted the overall learning curves for the data from these conditions, see Figure 2. We can see from this figure that BKT

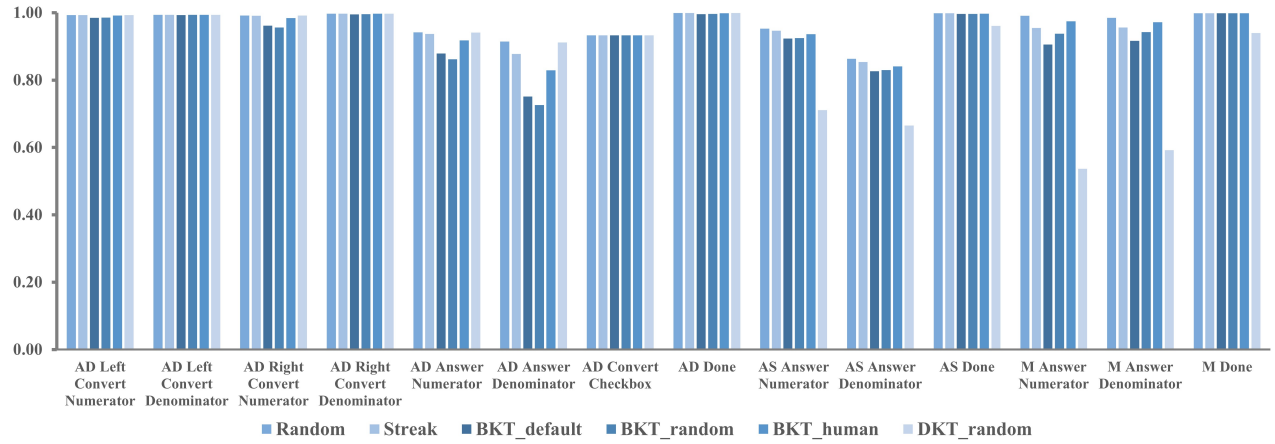


Figure 3: The AFM+S predicted probability at the end of training for each KC averaged over all students.

stops giving practice earlier than Streak, which subsequently stops giving practice earlier than Random and DKT. We observe higher variance in the tail of the learning curves for BKT and streak because the total number of students is decreasing as each student reaches mastery.

We applied the AFM+S model to predict performance on a hypothetical next opportunity for each KC and student. Figure 3 shows the average predicted correctness (across students) after the final practice opportunity for each skill. For most KCs, the prediction is higher than 95%, which suggests that mastery has been obtained in these KCs. Unfortunately the KC "AD Answer Denominator" has the lowest overall next-step correctness prediction in all six conditions. Figure 4 displays the learning curve for this skill across all six conditions and the number of students that have not yet mastered the skill at each point. This graph shows that there is a high slipping rate for this particular skill (see green line), indicating that there is a ceiling on the best possible AFM+S prediction that can be achieved.

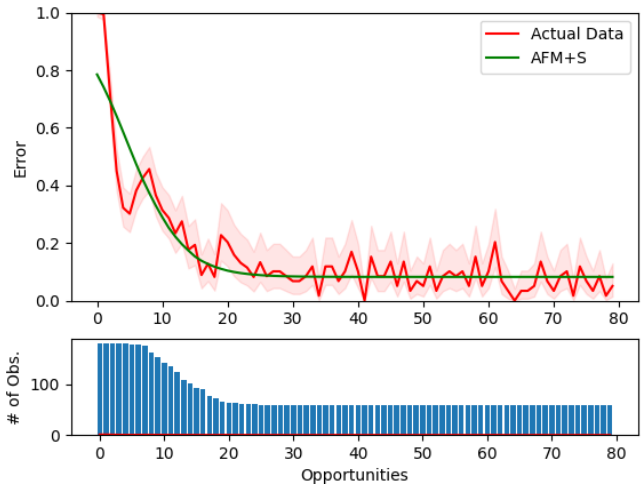


Figure 4: Learning curve for "AD Answer Denominator" and number of unmastered students at each opportunity.

Conditions	AD Average	AS Average	M Average
Random	0.01	0.01	0.01
Streak	0.06	0.11	0.12
BKT_default	0.09	0.16	0.16
BKT_random	0.10	0.17	0.15
BKT_human	0.08	0.16	0.12
DKT_random	0.01	0.30	0.31

Table 1: Model efficiency scores across six conditions.

To evaluate how well each approach handles the trade off between maximizing student's performance while minimizing the amount of practice, we computed a metric that we call "efficiency", see Table 1. To compute the score, we divided the AFM+S predictions at the end of training by the number of opportunities the student received for each student and KC. We then averaged over students to get a score for each KC. Finally, we averaged over KCs within each problem type. This produced, 3 model efficiency scores for each of the six models. Bigger value refers to a more efficient model. The efficiency score complements accuracy and provides more information for selecting the best model.

Although Random gives the highest prediction in Figure 3 among all KCs, it is the least efficient one as it gives all the problems during training. BKT_random has lower predictions than Streak, however the model efficiency suggests that it is more efficient. DKT_random yields the same efficiency as Random in AD problems. However, for AS and M problems, it appears to have the highest efficiency across all six models since it takes the least practice, but still achieves a moderate correctness prediction.

4.2 Simulated vs. Human BKT parameters

To validate the feasibility of generating BKT parameters using simulated student data, we did a correlation analysis of the BKT_random and BKT_human parameters. Figure 5 shows that there's a positive correlation of around 0.65 in the "Learn" parameter, which means the simulated students generated by Apprentice Learner have a similar learning rate as human students. We argue that this is one of the

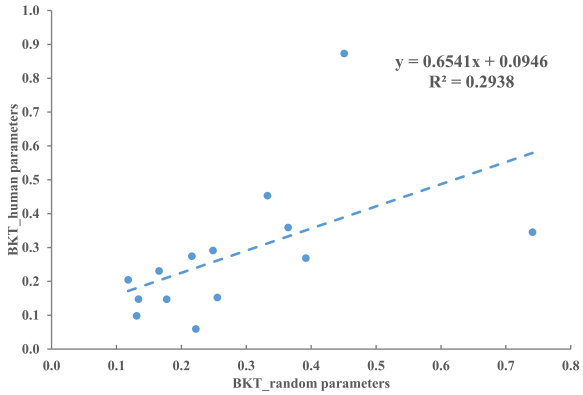


Figure 5: Positive correlation in the BKT "Learn" parameter values estimated from simulated and human data.

harder parameters to set. The "Known" parameter ($P(L_0)$) was near 0 for all skills in the simulated data because all agents start off without any prior knowledge. In human data, this parameter will vary based on the learning context. The "Guess" and "Slip" parameters based on the simulated data were reasonable (both greater than 0), but exhibited no notable correlation with the human guess and slip values. Taken together, we argue that this approach is a promising way to identify initial parameter values for BKT, but more research is needed to explore and generalize this idea.

4.3 Knowledge Tracing Sequence Analysis

Next, we take a closer look at the estimates of several knowledge tracing approaches for different sequences. Figure 6 shows the correctness sequence of a single student on "AD Answer Denominator" KC. This student was taken from the BKT_random condition. We added the mastery predictions generated by BKT, Streak and DKT given this sequence. For BKT and Streak, only the correctness on this skill was used. For DKT, the model was given the entire student sequence for all KCs, but only the predictions for this KC are shown. Predictions were taken at where the student just finished the problem that contained the target KC.

For BKT, the estimates trend towards increased mastery over the course of practice, but sometimes the probability decreases when it gets an item wrong. For Streak, each correct response yields a 33% increase in the mastery prediction, accumulating to 100% by the third correct response in a row. The DKT models predictions tend to jump around, but generally do not seem to be increasing despite getting the problem correct multiple times in a row. For example, the model's probability of correct jumps to 100% before returning to and staying close to 0%.

Correctness		✗	✗	✗	✓	✓	✓	✗	✓
BKT Mastery	0%	13%	16%	16%	72%	97%	100%	99%	100%
Streak Mastery	0%	0%	0%	0%	33%	67%	100%	0%	33%
DKT Mastery	36%	0%	0%	100%	0%	0%	0%	2%	0%

Figure 6: Different Model Predictions on "AD Answer Denominator" given student correctness sequence.

To figure out why the DKT model has such erratic behavior and why it gives so many AD problems, we fed a complete sequence from one student into the DKT model (student from BKT_random condition). Figure 7 shows the prediction of mastery for each KC after the student has completed each problem (problem type shown on the x-axis). It seems that the student never masters the "AD Answer Numerator" or "AD Answer Denominator", which explains why the DKT tutor is giving almost all the AD problems to the students.

Upon further investigation, we discovered that the DKT model has a fundamental issue that makes it difficult to use for mastery learning. The issue is caused by using the DKT predictions between problems when the mastery learning system is determining which KCs are mastered before picking another problem. Unfortunately, for multi-step problems some KCs cannot be correctly applied on the first step. DKT correctly predicts these KCs will have near 0% correctness (any attempts will be incorrect). However, this has the side effect of confusing the mastery learning system into thinking that the KC is unmastered. When the DKT model actually reaches a step where the KC can be correctly applied, then its predicted probability jumps to a more realistic estimate of the mastery. This problem was not identified in previous work on mastery learning with DKT (e.g., [15]) because the prior work only looked at problems with a single step, so this issue never occurred. However, most problems within tutoring systems are multi-step. Future work should explore how to correct this issue within DKT so it can be used for mastery learning.

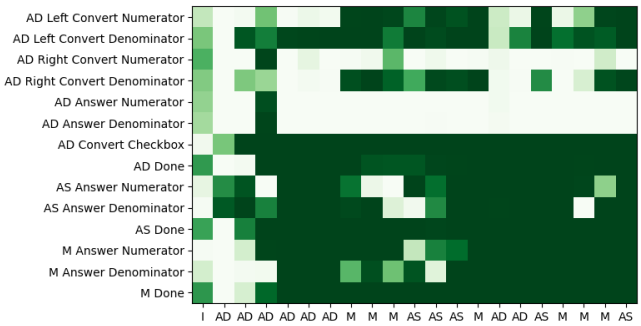


Figure 7: DKT predictions for each KC after each problem.

5. SUMMARY OF KEY FINDINGS

Our first key finding is that simulated students can successfully evaluate online knowledge tracing models. Our simulations indicate that Streak works well for mastery learning and has reasonable efficiency; although the model gives a bit more practice than strictly necessary (e.g. the average number of steps to master all AD KCs is around 17 in Streak and 10 in BKT_random). Still, Streak is very simple to operate, implement, and modify and it behaves reasonably well.

BKT also work well for mastery learning and generally seems to have the best efficiency of the approaches we compared (although DKT seems to be more efficient for AS and M problems). However, it seems to stop a little early in some cases, resulting in under practice. Figure 3 shows that the BKT_random model gets 86% correctness prediction for the KC "AD Answer Numerator", 73% for "AD Answer Denom-

inator" and 83% for "AS Answer Denominator". To some extent, it reveals that these KCs might be more difficult for both simulated students and human students to master, and more practice is required to obtain mastery. Our work suggests that it is important to look at multiple factors when evaluating knowledge tracing approaches in online settings. In particular, it is important to look at both student mastery and the amount of practice that is administered. To explore this trade off, we proposed the efficiency metric. We believe that this metric (or ones like it) might be useful for evaluating knowledge tracing approaches in online settings.

Multiple studies [18, 17] suggest that the DKT model has good predictive performance in offline setting. However, we found that it does not seem to work properly for online knowledge tracing, particularly in cases of multi-step problems. Yeung & Yeung [19] identified that DKT's predictions for KCs are not consistent across time-steps, which we believe is related to the issue we encountered. Even when a student performs well on a KC, DKT's predicted performance for that skill may drop. In general, DKT's predictions fluctuate drastically over time, so *when* its predictions are sampled has a big impact on its accuracy. The core issue is that DKT generates predictions for every KC at every step, but its loss function only constrains predictions that are likely to occur next. Yeung & Yeung suggested some possible modifications to the DKT objective function to mitigate this problem, but implementation and testing of these was beyond the scope of the current study.

In preliminary analysis, BKT estimated students reached mastery even though their error rates were still high. Further inspection revealed an error in our BKT model that was causing it to incorrectly estimate student mastery. Multiple researchers across multiple labs have used this open-source implementation. Despite wide use, we uncovered issues that had not been previously uncovered. Although we corrected these issues for the current study, we argue that this is a positive outcome for our simulated student approach. This finding reinforces the idea that simulated students can be used to test and improve knowledge tracing approaches before running more costly human studies.

Our second key finding is that researchers might use data generated by simulated students to initialize knowledge tracing parameters when human data is not available. To evaluate the feasibility of this idea for BKT, we conducted a correlation analysis between the random-based BKT parameters and human-based parameters. We found a strong correlation between the learning rate parameters suggesting that initializing BKT parameters using simulated student data might be an informed, but cost-effective approach.

6. RELATED WORKS

The closest work to ours is the simulation studies conducted by Doroudi et al. [4, 3], which investigates different knowledge tracing approaches using simulated students. They argue that it is important to evaluate knowledge tracing under various assumptions about how students learn. One of the major differences between their approach and ours is that they use statistical models that predict correctness to simulate students rather than computational models of learning that actually learn and perform the task, as we do with Apprentice agents. Apprentice agents are more complex than the knowledge tracing approaches that are being used to evaluate them. It would be interesting to explore the use of Apprentice agents as another kind of student model for the knowledge tracing evaluations proposed by Doroudi et al.

7. CONCLUSIONS AND FUTURE WORK

We were able to successfully apply simulated students to test different knowledge tracing models. When we compared the three knowledge tracing models (BKT, Streak, and DKT) to a no-knowledge-tracing baseline (Random), we found that BKT gave the fewest problems, Streak gave the second fewest, Random gave the most and DKT gave almost as many as Random in one problem type and the least in the other two. In general, we found that BKT seemed to be the most efficient approach, but streak gave reasonable results despite its simplicity. Through the use of simulated students, we also discovered a number of issues with our BKT implementation as well a fundamental issue with DKT. Despite widespread use of the BKT implementation and a lot of recent investigation into the DKT model, these issues had not been discovered in prior work. Together, these results support our primarily claim that simulated students are an effective tool for investigating and evaluating online knowledge tracing approaches.

Our analysis also found evidence to support the idea that simulated student data might be used to initialize BKT parameters when no human-student data is available. In particular, we found that BKT learning rates estimated from simulated data have a significant correlation to the learning rates estimated from human data. While these initial results are promising, more work is needed to further explore these ideas. In particular, we would like to try running human-subject experiments to compare BKT models initialized using simulated student data to those with default parameters. One surprising finding is how well BKT_default performs; despite somewhat arbitrary parameters, it was more efficient than Streak. Future work should explore how to manually pick robust default values for BKT.

We have a number of additional future directions we would like to explore. We intend to individualize the Apprentice models to make them better mimic the behaviors of different kinds of learners (e.g., high vs. low performing students), students with different motivation in learning, and those who suffer from learning disabilities. We should also explore variations of DKT that address concerns we have identified and enable its use in online mastery learning. Finally, we should move beyond simulation and explore how well our simulated students predict which knowledge tracing approaches will yield the best learning for human students.

8. ACKNOWLEDGMENTS

We would like to thank Danny Weitekamp and Erik Harpstead, who developed much of the framework for testing the Apprentice agents within the fractions tutors. We also thank Anna Raffery for creating the framework for applying knowledge tracing to simulated students and developing the initial version of Bayesian Knowledge Tracing that we used. We also thank Adit Gupta for reading earlier drafts and providing suggestions for improvement.

References

[1] H. Cen. *Generalized Learning Factors Analysis: Improving Cognitive Models with Machine Learning*. PhD thesis, Carnegie Mellon University Pittsburgh, 2009.

[2] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[3] S. Doroudi, V. Aleven, and E. Brunskill. Robust evaluation matrix: Towards a more principled offline exploration of instructional policies. In *Proceedings of the Fourth ACM Conference on Learning@Scale*, pages 3–12, 2017.

[4] S. Doroudi and E. Brunskill. Fairer but not fair enough on the equitability of knowledge tracing. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, pages 335–339, 2019.

[5] S. E. Fancsali, T. Nixon, and S. Ritter. Optimal and worst-case performance of mastery learning assessment with Bayesian knowledge tracing. *Proceedings of the 6th International Conference on Educational Data Mining, EDM 2013*, 2013.

[6] T. Gervet, K. Koedinger, J. Schneider, T. Mitchell, et al. When is deep learning the best approach to knowledge tracing? *Journal of Educational Data Mining*, 12(3):31–54, 2020.

[7] N. T. Heffernan and C. L. Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4):470–497, 2014.

[8] M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing? *Proceedings of the 9th International Conference on Educational Data Mining, EDM 2016*, pages 94–101, 2016.

[9] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the edm community: The pslc datashop. In *Handbook of Educational Data Mining*, volume 43, pages 43–56. CRC Press, 2010.

[10] C. J. MacLellan, E. Harpstead, R. Patel, and K. R. Koedinger. The apprentice learner architecture: Closing the loop between learning theory and educational data. *Proceedings of the 9th International Conference on Educational Data Mining, EDM 2016*, pages 151–158, 2016.

[11] C. J. MacLellan and K. R. Koedinger. Domain-general tutor authoring with apprentice learner models. *International Journal of Artificial Intelligence in Education*, pages 1–42, 2020.

[12] C. J. MacLellan, R. Liu, and K. R. Koedinger. Accounting for Slipping and Other False Negatives in Logistic Models of Student Learning. *Proceeding of the 8th International Conference on Educational Data Mining, EDM15*, pages 53–60, 2015.

[13] P. Nedungadi and M. S. Remya. Predicting students' performance on intelligent tutoring system - Personalized clustered BKT (PC-BKT) model. *Proceedings - Frontiers in Education Conference, FIE*, 2015-February(February), 2015.

[14] R. Patel, R. Liu, and K. R. Koedinger. When to block versus interleave practice? evidence against teaching fraction addition before fraction multiplication. In *Proceedings of Cognitive Science Conference*, 2016.

[15] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 505–513, Cambridge, MA, USA, 2015. MIT Press.

[16] J. Rollinson and E. Brunskill. From predictive models to instructional policies. *Proceedings of the 8th International Conference on Educational Data Mining, EDM 2015*, pages 179–186, 2015.

[17] L. Wang, A. Sy, L. Liu, and C. Piech. Deep knowledge tracing on programming exercises. In *Proceedings of the Fourth ACM Conference on Learning@Scale*, pages 201–204, 2017.

[18] X. Xiong, S. Zhao, E. G. Van Inwegen, and J. E. Beck. Going deeper with deep knowledge tracing. *Proceeding of the 9th International Conference on Educational Data Mining, EDM 2016*, pages 545–550, 2016.

[19] C.-K. Yeung and D.-Y. Yeung. Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proceedings of the Fifth Annual ACM Conference on Learning@Scale*, pages 1–10, 2018.

[20] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *Lecture Notes in Computer Science*, volume 7926 LNAI, pages 171–180, 2013.

Effects of Algorithmic Transparency in Bayesian Knowledge Tracing on Trust and Perceived Accuracy

Kimberly Williamson
Cornell University
Ithaca, NY, USA
khw44@cornell.edu

René F. Kizilcec
Cornell University
Ithaca, NY, USA
kizilcec@cornell.edu

ABSTRACT

Knowledge tracing algorithms such as Bayesian Knowledge Tracing (BKT) can provide students and teachers with helpful information about their progress towards learning objectives. Despite the popularity of BKT in the research community, the algorithm is not widely adopted in educational practice. This may be due to skepticism from users and uncertainty over how to explain BKT to them to foster trust. We conducted a pre-registered 2x2 survey experiment (n=170) to investigate attitudes towards BKT and how they are affected by verbal and visual explanations of the algorithm. We find that ostensible learners prefer BKT over a simpler algorithm, rating BKT as more trustworthy, accurate, and sophisticated. Providing verbal and visual explanations of BKT improved confidence in the learning application, trust in BKT and its perceived accuracy. Findings suggest that people's acceptance of BKT may be higher than anticipated, especially when explanations are provided.

Keywords

Bayesian Knowledge Tracing, Data Visualization, Explainable AI

1. INTRODUCTION

Knowledge tracing can offer students and teachers a real-time understanding of what students have already learned and what they are still struggling with [7]. It provides actionable insights that can lead to better educational outcomes [16]. Among many types of knowledge tracing algorithms, Bayesian Knowledge Tracing (BKT) has been established and researched most extensively, as evidenced by the 114,000 Google Scholar results for "Bayesian Knowledge Tracing," 17,500 of which published since 2020. BKT has been tested to help students self-monitor their learning progress [4, 23], to help teachers understand what students have not learned yet [22], and to enable adaptive learning technologies that let students skip over the content they have mastered [18]. In contrast to the abundance of research on BKT, including hundreds of articles devoted to incremental enhancements of the original model [20], there are not many real-world applications that use BKT in practice. Some of the most widely used K-12 learning platforms like ASSISTments and Khan Academy decided against using BKT in favor of simpler models such as N-Consecutive Correct Responses (N-CCR) [13]. This raises questions about barriers to adopting knowledge tracing algorithms in educational practice. In particular, how much is the relative complexity and opacity of BKT responsible for its slow adoption? Platform providers may be concerned that educators and learners will not trust a model that cannot easily be explained to them [13, 12, 24, 25, 1].

The Technology Acceptance Model (TAM) posits that a user's acceptance and adoption of new technology is based on its perceived usefulness (PU) and perceived ease of use (PEOU) [9]. PU and PEOU are beliefs that can be influenced by external factors, such as providing additional information about a technology. According to TAM, learners' and educators' PU and PEOU are essential factors in the adoption of BKT in practice. Improving their perceptions could therefore increase the acceptance and adoption of BKT in real-world applications. Moreover, a better understanding of the mechanisms behind the acceptance of BKT is expected to inform the presentation of other knowledge tracing algorithms as well.

A large number of knowledge tracing algorithms have been developed over the years that could benefit from empirical evidence on how to explain them to users. Recent advances in artificial intelligence have inspired research into more complex algorithms such as deep knowledge tracing (DKT), which uses neural networks [17, 11]. With more complex algorithms that provide less insight into their inner workings, it becomes more important to understand how people's trust in the algorithm and its perceived accuracy might influence perceptions of usefulness and usability of a learning application [1]. Besides BKT and DKT, which are suitable for modeling understanding and sense-making, there are also logistic learning models, such as Additive Factor Models and Performance Factor Analysis [5, 19, 20], which model memory and fluency [20]. These two types of models can also be integrated into one [15]. While there are many types of models that can be examined, we choose BKT as an example knowledge tracing algorithms that is relatively simple and popular among researchers.

Kimberly Williamson and Rene Kizilcec "Effects of Algorithmic Transparency in Bayesian Knowledge Tracing on Trust and Perceived Accuracy". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 338-344. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

This research contributes causal evidence to address three important research questions. First, do people prefer to learn with BKT or N-CCR (N-Consecutive Correct Responses) in an ostensible high-stakes test scenario? Second, how is their preference related to specific attitudes, including their confidence in the learning system to do well on a test, their trust in the algorithm, and the perceived accuracy of the algorithm? And third, how do verbal and/or visual explanations affect people's attitudes and preferences over knowledge tracing algorithms? We answer these research questions with data collected from a pre-registered 2x2 factorial survey experiment.

2. BACKGROUND

One of the simplest knowledge tracing algorithms is N-CCR. It assesses student mastery by evaluating the number of consecutive correct responses for a particular skill. For example, the model determines that a student has learned fractions after correctly answering three fraction questions in a row. Although N-CCR is easy to understand, its simplicity can sometimes make it less accurate than BKT. Still, N-CCR has been used in popular platforms, including ASSISTments and Khan Academy [13], and there is mixed evidence as to whether BKT outperforms N-CCR at modeling student learning [8, 10, 13, 21]. Nevertheless, the scientific community shows a clear preference for BKT (and other more complex knowledge tracing algorithms) based on the allocation of research attention.

BKT is a two-state Hidden Markov Model where the unobserved hidden state being modeled is student learning, and for a given knowledge component, a student has a state of either learned or not learned [6, 17, 11]. Although BKT is already more sophisticated than N-CCR, critics have suggested that BKT is too simple of an algorithm for modeling human learning. They point to deep (neural network) learning models to better represent all factors that go into student learning [17, 11]. Mao and colleagues [17] found that deep learning models outperformed BKT on some learning tasks. However, they also acknowledge that these gains in performance might not be worth the loss in model interpretability. While researchers tend to consider BKT as one of the simpler and more explainable algorithms for knowledge tracing, practitioners and learners who are the end-users may not share this view.

The explainability of an algorithm, which is partly determined by how transparent, understandable, interpretable it is, can play an essential role in its adoption into applications. Barredo Arrieta and colleagues [1] identified these and other reasons for making algorithms more explainable: most relevant to the work on BKT are trustworthiness, confidence, causality, and accessibility. Prior research on algorithms in education has echoed this finding. Kizilcec [14] found that increasing transparency by providing users with additional information about an algorithm made users trust the algorithm more (though too much information can erode trust). Other studies have more specifically examined the interpretability of BKT in learning applications. Yeung [24] explored the use of Item Response Theory to make BKT and deep learning models more explainable, but they have not examined how users react to it. Zhou and colleagues [25] examined BKT explainability by creating visualization "explainables." They then designed an experiment to determine the effectiveness between a static and interactive visualization and found that the static explainable led to a better understanding of the BKT algorithm. More generally, research on Open Learning Models (OLMs) has advanced an understanding of how to visualize and explain learning models [3, 2]. OLMs provide users with interactive visualizations that grant them insights into learning algorithms, along with the ability to adjust the algorithm. This study will add to OLM research by expanding knowledge on how to explain and visualize information to foster positive attitudes.

The current study provides a foundational understanding of how individuals perceive BKT compared to N-CCR along several attitudinal dimensions, and how much verbal and visual explanations of BKT can improve those perceptions. Our review of prior work informed the following two hypotheses:

H1. Verbal and visual explanations of BKT lead participants to prefer it over N-CCR.

H2. Verbal and visual explanations of BKT will positively increase participants attitudes about the BKT algorithm.

3. METHODS

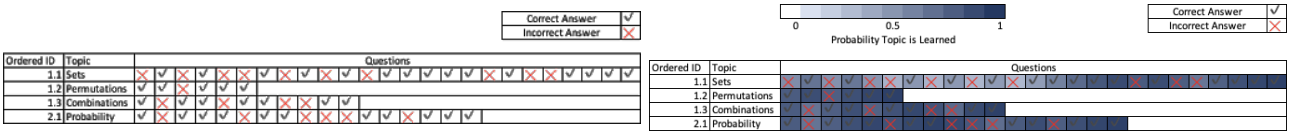
The study design, materials, measures and analysis approach are pre-registered with the Open Science Foundation: https://osf.io/7c5zt/. To refine the study design, measures, and analysis plan, we ran a pilot study with 26 participants and used both descriptive and inferential statistical analyses to build our analysis plan. We first used descriptive analysis to estimate survey completion time, ensure we had enough variance in responses, and check that the information provided to participants was enough information for them to evaluate the algorithms. We used respondents' answers and an open-ended question at the end of the survey in which we asked participants for any feedback to improve the survey. We took the results from this pilot study to alter the visualizations and information provided to participants and rephrase some questions to improve clarity. We removed the open-ended feedback question from the survey after the pilot.

3.1 Participants

Participants were recruited from Amazon Mechanical Turk and received $1.70 for completing a 10-minute survey. The study was advertised as seeking input on test preparation applications. To determine our target sample size of 170, we used G*Power to conduct a power analysis. Our analysis goals were to obtain 95% power to detect a medium effect size of 0.25 at the standard 0.05 alpha error rate with six repeated measures and four groups. While we had 170 participants who took the survey, 34 participants either failed to answer all of the comprehension questions correctly (29) or had prior experience with BKT (4) or both (1). Analyses were conducted on the remaining 136 respondents. Table 1 describes the sample demographics for the sample.

3.2 Procedure

To contextualize the study, participants were provided the following narrative with pictures of two sample questions taken from the ASSISTments platform:



(a) Simple visualization

(b) Detailed visualization

Figure 1: Two versions of a visualization of student performance on questions shown to participants depending on their condition assignment.

Table 1: Sociodemographics of Participants included in the study.

		n	%
Gender	Woman	79	58.1
	Man	54	39.7
	Transgender Man	1	.7
	Gender Variant/Non-Conforming	2	1.5
Ethnicity	Hispanic	10	7.4
	Not Hispanic	126	92.6
Race	White	100	73.5
	Black or African American	10	7.3
	American Indian or Alaska Native	2	1.5
	Asian	16	11.8
	Not Listed	5	3.7
	Multiracial	3	2.2
Age	18-24	27	19.9
	25-34	52	38.2
	35-44	37	27.2
	45-54	8	5.9
	55-64	11	8.1
	Above 65	1	.7

As an admissions requirement for a university program that you are applying for, you are preparing to take a general knowledge exam. The test is important to you and you need to do as well as possible to get accepted.
You have decided to use a test preparation app to help you study for the test.
A key feature of the test prep app is that it ***personalizes*** *the learning experience to help you study efficiently. The app shows you* ***only questions about topics that you have not already learned.***
The system keeps track of your answers to each question and ***automatically moves to the next topic once it determines that you have learned the previous topic.*** *To determine if you have learned a topic, the app uses an algorithm. Once the* ***algorithm determines you learned a topic****, it will stop giving you study questions about it. Thus, it also determines the speed at which you progress in your test prep.*
We would like to get your opinions about the ***two different algorithms*** *to understand which one you find more accurate and trustworthy.*

On the next page, participants answered three multiple-choice comprehension questions: (1) How does the test prep app determine what questions to give you? (2) What determines how quickly you are going to be done with test prep? (3) What happens when the system determines that you have learned a topic? We pre-tested these questions to ensure that an attentive reader would have no problems answering them correctly.

Next, participants saw a short description of the N-CCR algorithm, which we labeled as 3 Right in a Row (3RR): "A topic will be considered learned once a student correctly answers three questions in a row." A simple table depicting a sample student's progression for four topics (table rows) and questions for each topic (table columns) accompanied the description. The table looked like Figure 1a. Each cell contained an X or a ✓depending on if the student answered the question correctly. True to the 3RR algorithm, each topic was considered learned once three consecutive questions were answered correctly. At the bottom of the page, participants answered several questions about their attitudes towards the 3RR algorithm (see Measures).

At this point, participants were randomly assigned to conditions based on a 2x2 factorial design. There were 33 participants in the No BKT Explanation/BKT Simple Visualization condition in the final sample, 34 in the No BKT Explanation/BKT Detailed Visualization condition, 38 in the BKT Explanation/BKT Simple Visualization condition, and 31 in the BKT Explanation/BKT Detailed Visualization condition.

The following page mirrored the structure of the previous one but for BKT, providing a description and sample learning progress visualization based on the experimental assignment, followed by the same set of attitudinal questions about the algorithm. Next, on the final page of the survey, participants were asked to compare the two algorithms.

3.3 Experimental Manipulations

In the no BKT explanation condition, participants received this one-sentence description of the BKT algorithm: "A topic will be considered learned once the algorithm estimates with a high probability that a student has learned the topic based on their responses up to that point." In the BKT explanation condition, participants additionally received the following information about the BKT algorithm:

After every question you answer, the Bayesian Knowledge Tracing algorithm estimates the probability that you have now learned a topic using a probabilistic model that accounts for the following data:
– an initial probability that you have learned the topic based on your first answer: it is higher if you answered correctly
– a correct guess probability: e.g., 50% for a true/false question
– a slip probability for answering incorrectly even though you already learned the topic

– the difficulty of questions you have answered based on how many people have answered them incorrectly
– performance data such as the number of hints that you asked for and the time it took you to answer the question

Using all of this information, the algorithm estimates the probability that you have learned a topic. If the probability is above 95%, the algorithm moves you on to the next topic.

In the BKT simple visualization condition, participants received a simple table depicting a sample student's learning progress mirroring the one shown in the 3RR algorithm (Figure 1a). In the BKT detailed visualization condition, the same table was enhanced to show the estimated probability of having learned the topic using a color scale (Figure 1b). To make the visualizations realistic, we ran a BKT algorithm over a sample of ASSISTments data and used a 95% probability to determine mastery.

3.4 Measures

We measured participants' attitudes towards each algorithm using six items rated on 5-point unipolar response scales ('Not at all', 'Somewhat', 'Moderately', 'Very', 'Extremely'):

Confidence: "How confident are you that the test prep app with this algorithm will prepare you to do very well on the test?"
Understanding: "How well do you understand how this algorithm determines if you have learned a topic?"
Sophistication: "How complex is this algorithm for determining if you have learned a topic?"
Accuracy: "How accurate is this algorithm at determining if you have learned a topic?"
Trust: "How much do you trust this algorithm to determine what you have learned?"
Speed: "How quickly do you learn the materials for the test using this algorithm?"

At the end of the survey, participants rated their general preference over the two algorithms in response to the following question: "Now that you have learned about the 3 Right in a Row (3RR) and Bayesian Knowledge Tracing (BKT) algorithms, which one would you prefer to use for your test prep?" Response options were on a 7-point bipolar scale: 'Strongly prefer 3RR', 'Moderately prefer 3RR', 'Slightly prefer 3RR', 'Neither prefer 3RR nor BKT', 'Slightly prefer BKT', 'Moderately prefer BKT', 'Strongly prefer BKT'. Participants were invited to provide a rationale for their preference using an open-ended question: "Please tell us why you prefer the algorithm that you choose above."

3.5 Analytical Approach

We used the pilot study data to finalize our analysis plan by developing our inferential analysis. For H1, we decided to use linear regression to understand if the conditions had an association effect on the participants' overall preference. We used the conditions as the predictor variables and the preference as the outcome variable. We next decided to use multiple linear regression to understand the association between the attitudinal constructs and algorithm preferences. This analysis used the attitudinal constructs as the predictor variables with preference as the outcome variable. The last planned analysis evaluated H2 by running a linear regression on each attitudinal construct with the conditions as the predictor variables and the attitudinal construct as the dependent variable. While the interpretation of linear regression output is clear and familiar, we acknowledge that our measures are ordinal and not strictly continuous. We confirmed that analysis by ordinal logistic regression yields equivalent results.

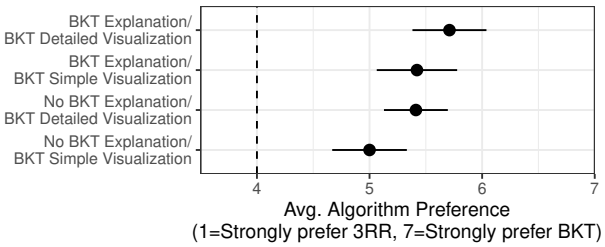


Figure 2: Average algorithm preference by condition.

For the open-ended question asking participants why they choose their preferred algorithm, we planned to use simple thematic coding. While we used the pilot data to create our analysis plan, we did remove all pilot data from the final dataset.

4. FINDINGS

First, we examine which algorithm participants preferred overall. Figure 2 shows their average preference in each condition, which varied between 5 (i.e. Slightly prefer BKT) and 6 (i.e. Moderately prefer BKT). While there is a suggestive pattern that providing more explanation for BKT strengthens the preference for BKT, this pattern was not statistically significant (linear regression: $F_{3,132} = 0.7455, p = 0.5268$). This means the data do not support H1.

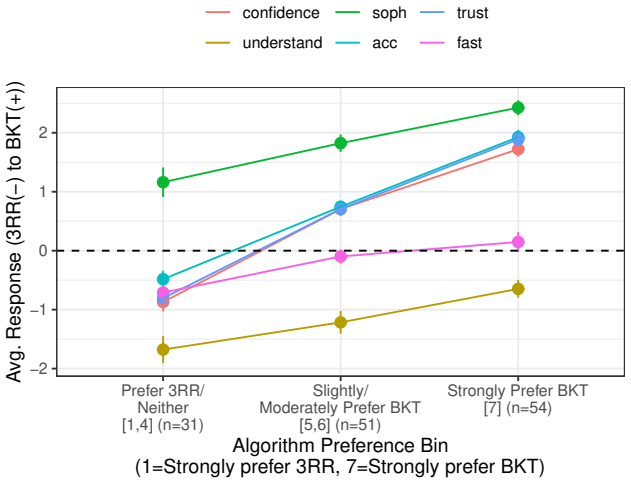


Figure 3: Average response on each measure at three levels of preference: Prefer 3RR to Neither, Slightly and Moderately Prefer BKT, and Strongly Prefer BKT. We choose these groupings because each group represents approximately 1/3 of the sample.

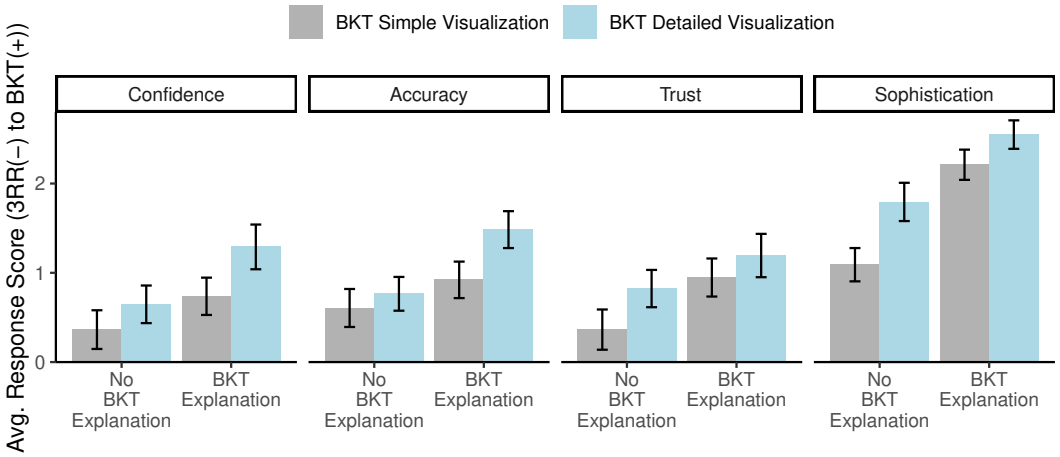


Figure 4: Average differences (BKT score - 3RR score) in significant attitudinal constructs as a function of the randomly assigned conditions. Positive scores indicate a higher score for BKT.

Next, we examine how algorithm preference is related to the six attitudinal measures: confidence in the learning system, the sophistication of the algorithm, trust, understanding, accuracy, and speed. We use the repeated measures design of our study by computing the difference score for each question: subtracting the participant's 3RR response from the BKT response. Figure 3 shows the average response on each measure at three levels of preference: Prefer 3RR to Neither, Slightly and Moderately Prefer BKT, and Strongly Prefer BKT. We choose these grouping because each group represents approximately 1/3 of the sample. All measures are positively correlated with preference as evidenced by their positive slopes (all Pearson's $r > 0.325, p < 0.0001$), but accuracy, confidence, and trust are correlated more strongly ($r > 0.739$). This highlights the importance of these three constructs in determining people's preference over the algorithms. In fact, the six measures explain 66.7% of the variance in preferences (multiple linear regression: $F_{6,129} = 43.07, p < 0.0001$).

Lastly, we examine how the provision of verbal and/or visual explanations influenced participant attitudes about BKT. Figure 4 shows the average response in each condition for the four measures that were significantly affected by the intervention (i.e., relative understanding and speed did not change significantly at $p < 0.1$). We find that confidence in the learning application with BKT (relative to 3RR) improved when both a detailed explanation and visualization were provided ($F_{3,132} = 2.88, p = 0.03844$). Likewise, the perceived accuracy of BKT improved with both types of explanation provided ($F_{3,132} = 3.28, p = 0.02305$). Trust in BKT improved by providing a detailed explanation, especially when complemented with the detailed visualization ($F_{3,132} = 2.346, p = 0.07575$). Finally, and not surprisingly, the more detail was provided, the more sophisticated BKT was perceived to be ($F_{3,132} = 11.17, p < 0.001$). This provides evidence in support of H2.

In all of our analyses, we tested for the presence of demographic heterogeneity in results. However, no significant demographic sources of variation were found in our sample.

5. DISCUSSIONS

This study investigated people's attitudes towards BKT relative to a more straightforward knowledge tracing algorithm and tested the effect of additional information via explanations and visualizations on their attitudes. Understanding how students might perceive the algorithms used in their learning applications is a crucial issue for the adoption and usability of these tools [9]. The results provide evidence supporting our second hypothesis that additional explanations improve key attitudinal measures of confidence, perceived accuracy, trust, and sophistication. Qualitative data from participants echo this result:

> For something high stake, I'd only trust the methods that employs a variety of learning modalities. The analytics for such should match the complexities of my learning process as well as the nature of the material I'm learning. The BKT would put me more at ease than the quick route of the 3RR approach. (Participant assigned to BKT Explanation and BKT Detailed Visualization who had high confidence, sophistication, accuracy, and trust in BKT relative to 3RR)

Surprisingly, we did not find a significant increase in people's preference for BKT (H1), even though we found that algorithm preference is explained largely by people's perceptions of accuracy, trust, and confidence. This preference for BKT regardless of experimental condition is furthered explained by the qualitative responses from participants:

> I don't believe the 3RR algorithm is at all beneficial to the student attempting to learn the topic. If the student just happens to get 3 exceptionally easy questions in a row, the algorithm will assume that the student has learned the topic which is likely not entirely true. (Participant assigned to No BKT Explanation and BKT Simple Visualization who Strongly Preferred BKT)

Nevertheless, participants generally preferred to use the BKT algorithm regardless of the experimental treatment.

Since confidence, trust, and accuracy are important to a user's preference for BKT, it was notable that those three measures were affected by the experimental manipulations. Consistent with prior studies of explainability and transparency in algorithmic systems, we also found that when more information about an algorithm is presented to people, they believe the algorithms to be more trustworthy and accurate, leading the user to have more confidence in the algorithm [1, 14]. This confidence can, in turn, increase the use of applications shown to improve educational outcomes. Our results further highlight the importance of OLM to provide more transparency to gain user trust and confidence. In addition, future educational applications using complex knowledge tracing algorithms should include detailed verbal and visualization explanations of the algorithm to improve confidence and trust in the application.

Frankly, we did not expect to find such strong support for BKT going into this study. Many learning platforms that have the capacity to implement BKT or even more complex algorithms have opted not to do so. Our informal understanding was that this was largely due to concerns that users, such as math teachers who use ASSISTments to assign homework, would not understand BKT and not trust it and lose confidence in the platform. This understanding led us to expect that participants would report a preference for the simpler algorithm and report that they find it more trustworthy and have more confidence in a system that uses it. Therefore, we are surprised by our positive findings for BKT and propose future research directions to follow up on these results.

Future work in this area should explore different participant populations and scenarios. We are planning to run this study with student samples to see if we can replicate the results. While we asked our participants to put themselves in the shoes of a student needing to prepare for a high-stakes test, running this experiment on a student population might make the scenario more realistic to the participants. Additionally, given that many tasks on Mechanical Turk involve subjects annotating machine learning datasets, this group of participants may have more favorable attitudes to algorithms. Another area for future work includes changing the scenario. We ran the study from the viewpoint of a student. However, we acknowledge that intelligent tutoring systems deployed in classrooms also need the trust and confidence of the teachers administering the applications. We plan to rewrite the scenario to allow participants to take on the role of a teacher.

6. REFERENCES

[1] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58(October 2019):82–115, jun 2020.

[2] S. Bull. There are open learner models about! *IEEE Transactions on Learning Technologies*, 13(2):425–448, 2020.

[3] S. Bull and J. Kay. Open learner models. In *Advances in intelligent tutoring systems*, pages 301–322. Springer, 2010.

[4] A. Bunt and C. Conati. Probabilistic student modelling to improve exploratory behaviour. *User Modeling and User-Adapted Interaction*, 13(3):269–309, 2003.

[5] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International Conference on Intelligent Tutoring Systems*, pages 164–175. Springer, 2006.

[6] B. Choffin, F. Popineau, Y. Bourda, and J.-J. Vie. Das3h: Modeling student learning and forgetting for optimally scheduling distributed practice of skills. In *International Conference on Educational Data Mining (EDM 2019)*, 2019.

[7] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[8] Y. B. David, A. Segal, and Y. K. Gal. Sequencing educational content in classrooms using Bayesian knowledge tracing. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge - LAK '16*, pages 354–363, New York, New York, USA, 2016. ACM Press.

[9] F. D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340, 1989.

[10] S. Doroudi and E. Brunskill. Fairer but Not Fair Enough On the Equitability of Knowledge Tracing. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, pages 335–339, New York, NY, USA, mar 2019. ACM.

[11] T. Gervet, K. Koedinger, J. Schneider, T. Mitchell, et al. When is deep learning the best approach to knowledge tracing? *Journal of Educational Data Mining*, 12(3):31–54, 2020.

[12] W. J. Hawkins, N. T. Heffernan, and R. S. J. D. Baker. Learning bayesian knowledge tracing parameters with a knowledge heuristic and empirical probabilities. In S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia, editors, *Intelligent Tutoring Systems*, pages 150–155, Cham, 2014. Springer International Publishing.

[13] K. Kelly, Y. Wang, T. Thompson, and N. Heffernan. Defining Mastery: Knowledge Tracing Versus N-Consecutive Correct Responses. In *8th International Conference on Educational Data Mining*, 2015.

[14] R. F. Kizilcec. How Much Information? In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2390–2395, New York, NY, USA, may 2016. ACM.

[15] S. Klingler, T. Käser, B. Solenthaler, and M. Gross. On the performance characteristics of latent-factor and knowledge tracing models. *International Educational Data Mining Society*, 2015.

[16] K. R. Koedinger and V. Aleven. Exploring the assistance dilemma in experiments with cognitive tutors. *Educational Psychology Review*, 19(3):239–264,

2007.

[17] Y. Mao. Deep learning vs. bayesian knowledge tracing: Student models for interventions. *Journal of educational data mining*, 10(2), 2018.

[18] E. Millán and J. L. Pérez-De-La-Cruz. A bayesian diagnostic algorithm for student modeling and its evaluation. *User Modeling and User-Adapted Interaction*, 12(2):281–330, 2002.

[19] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger. Performance factors analysis–a new alternative to knowledge tracing. *Online Submission*, 2009.

[20] R. Pelánek. Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*, 27(3):313–350, 2017.

[21] R. Pelánek and J. Řihák. Experimental Analysis of Mastery Learning Criteria. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 156–163, New York, NY, USA, jul 2017. ACM.

[22] S. Ritter, M. Yudelson, S. E. Fancsali, and S. R. Berman. How mastery learning works at scale. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pages 71–79, 2016.

[23] S. Schiaffino, P. Garcia, and A. Amandi. eteacher: Providing personalized assistance to e-learning students. *Computers & Education*, 51(4):1744–1754, 2008.

[24] C.-K. Yeung. Deep-IRT: Make Deep Learning Based Knowledge Tracing Explainable Using Item Response Theory. *EDM 2019 - Proceedings of the 12th International Conference on Educational Data Mining*, pages 683–686, apr 2019.

[25] T. Zhou, H. Sheng, and I. Howley. Assessing post-hoc explainability of the bkt algorithm. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, page 407–413, New York, NY, USA, 2020. Association for Computing Machinery.

Automatic short answer grading with SBERT on out-of-sample questions

Aubrey Condor
University of California, Berkeley
aubrey_condor@berkeley.edu

Max Litster
University of California, Berkeley
maxlitster@berkeley.edu

Zachary Pardos
University of California, Berkeley
pardos@berkeley.edu

ABSTRACT

We explore how different components of an Automatic Short Answer Grading (ASAG) model affect the model's ability to generalize to questions outside of those used for training. For supervised automatic grading models, human ratings are primarily used as ground truth labels. Producing such ratings can be resource heavy, as subject matter experts spend vast amounts of time carefully rating a sample of responses. Further, it is often the case that multiple raters must come to a census before a final ground-truth rating is established. If ASAG models were developed that could generalize to out-of-sample questions, educators may be able to quickly add new questions to an auto-graded assessment without a continued manual rating process. For this project we explore various methods for producing vector representations of student responses including state-of-the-art representation methods such as Sentence-BERT as well as more traditional approaches including Word2Vec and Bag-of-words. We experiment with including previously untapped question-related information within the model input, such as the question text, question context text, scoring rubric information and a question-bundle identifier. The out-of-sample generalizability of the model is examined with both a leave-one-question-out and leave-one-bundle-out evaluation method and compared against a typical student-level cross validation.

Keywords

ASAG, Assessment, SBERT, Generalizability

1. INTRODUCTION

Automatic Short Answer Grading (ASAG) is an emerging field of research, as the education community has started to embrace the use of technology to assist students and education professionals. It has been shown that the use of open-ended (OE) questions helps facilitate learning [7], but educators are often deterred from their use because grading requires much more time than that for multiple choice [12]. In addition, human ratings may contain bias and vary in consistency, as rating choices are often subjective [28]. ASAG systems may be an important tool for educators, allowing more frequent use of OE questions, and more objective ratings for both formative and summative assessments.

A key challenge with supervised automatic grading models is gathering a large enough sample of labelled data for training. While some labeling tasks for supervised learning may be straightforward such as identifying an image as either a dog or a cat, others such as rating student responses require careful consideration. In high-stakes assessment scenarios, two or more ratings by different experts are often necessary to form a reliable consensus rating. Thus, obtaining labelled data to train an ASAG system can be arduous. It follows that quickly introducing new questions to an existing system may not be feasible if a data collection as well as the meticulous rating of new responses is necessary. Further, a new model would have to be trained and tuned with the newly collected responses. If we create more generalizable ASAG models, educators may have the flexibility to add new questions to an existing assessment with very little effort, thus increasing the practical use of the ASAG system.

We hypothesize that the inclusion of extra question related information within the model input may improve both the classification performance, and the generalizability of the model. For the purposes of this project, we formally define the generalizability of an ASAG model as the capacity to classify responses from out-of-training-sample questions.

This research contributes to the field of automatic grading in three related ways. We focus on classification performance and generalizability of the supervised grading model in terms of 1) the textual representation type, 2) the content of the input and 3) the classification model. We compare three different representation types, including those of state-of-the-art models: Sentence-BERT, Word2Vec, and Bag of Words. In terms of input content, we experiment with including previously untapped resources relating to the questions in the model input. Such resources include a question-bundle identifier, the question stem text, question context text, and rubric information. Extra input content is vectorized (if the source is textual) and concatenated to the response vectors

Aubrey Condor, Max Litster and Zachary Pardos "Automatic short answer grading with SBERT on out-of-sample questions". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 345-352. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

to be used as input to the classification model. Finally, we compare a non-neural model (a multinomial logistic regression) and a simple neural (a three layer feed forward network) model.

In order to examine the generalizability of the model for each experiment, we use a leave-one-question-out evaluation procedure where we train the model on N-1 questions, and use the one left-out question data as our test set. Thus, during training, the model has not yet seen responses from the question for which we use solely to evaluate the model. We go one step further in testing the generalizability of the model with a leave-one-bundle-out evaluation procedure where we train the model on M-1 question bundles (groupings of questions that are related in context), and use the questions for the left-out bundle as the test set. We conceptualize the leave-one-bundle-out method as a more extreme test of the model's ability to classify out of sample questions because even questions that are related in context have not been seen by the model during training. Additionally, we compare results of our experiments against and a typical student level cross validation. Results from a majority class classifier are included as well for a baseline comparison.

2. RELATED WORK

This section outlines notable work relating to ASAG and the more general use of NLP for education. For this project, we build on the previous literature by considering lessons learned in preceding research, and employing novel approaches that, to our knowledge, have not yet been explored.

2.1 ASAG work

A systematic review of trends in ASAG [3] illustrates an increasing interest in the field of automatic grading for education. Unsupervised methods have been explored such as concept mapping, semantic similarity, and clustering to assign ratings. For example, Mohler and Mihalcea [19] compared knowledge based and corpus based semantic similarity measures for automatic grading, Klein et al. [14] implemented a latent semantic analysis approach, and Basu et al. [2] used clustering to provide rich feedback to groups of similar responses. In addition, many types of supervised classification methods have been utilized for ASAG. Noteable examples include Hou and Tsao [13] who incorporated POS tags and term frequency with a Support Vector Machine classifier, and Madnani et al. [16] who made use of simple features such as a count of commonly used words and length of response with a logistic regression classifier.

More recent ASAG research exploits deep learning methods. Noteable work includes Zhang et al. [31] who used a combination of feature engineering and deep belief networks, Liu et al. [15] who employed multi-way attention networks, and Yang et al. [30] who considered a deep autoencoder model specific to Chinese responses. Additionally, Qi et al. [21] created a hierarchical word-sentence model with a CNN and Bi-LSTM model and Tan et al. [25] explored the use of a graph convolutional network (GCN) to encode a graph of all student responses.

Further, much of the newest ASAG work makes use of state-of-the-art transformer based models, including Gaddipati et al. [11] who evaluated four different types of response embeddings, ELMo, GPT, BERT, and GPT-2 for their performance on an ASAG task, Camus and Filighera [4] who compared the performance of transformer models for ASAG in terms of the size of the transformer and the ability to generalize to other languages, and Sung et al. [24][23] who examined the effectiveness of pre-training BERT, including further pre-training the model on relevant domain texts.

2.2 NLP for Education

Literature addressing the general application of natural language processing (NLP) for various uses in field of education has grown quickly in recent years as well. For example, Fonseca et al. [10] used NLP to automatically classify the programming assignments for students within given academic context, Thaker et al. [26] incorporated textual similarity techniques to recommend remedial readings to students, and Arthurs and Alvero [1] examined bias in word representations for college admissions essays. Additionally, Xiao et al. [29] employed NLP and transfer learning methods for problem detection in peer assessments, Venant and d'Aquin [27] utilized a concept graph to predict semantic complexity of short essays by written by English language learners, and Chen et al. [6] leveraged a variety of textual analysis methods to predict student satisfaction in the context of online tutorial dialogues.

We build on the previous literature by incorporating state-of-the art representation methods such as Sentence-BERT and a neural classification model. The novel contribution of this project includes both our focus on the generalizability of the model to out-of-training-sample questions, as well as the leveraging of previously untapped, question related information as input to the model.

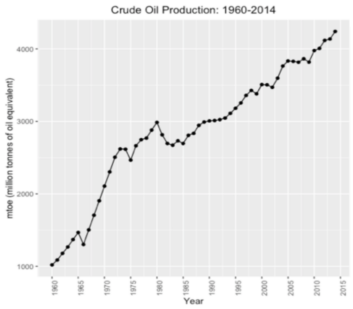
3. DATA SET

The data we will use for this project was sourced from a 2019 field test of a Critical Reasoning for College Readiness (CR4CR) assessment [17] created at the Berkeley Evaluation and Assessment Research (BEAR) center. The data consists of 5,550 student responses from 558 distinct students to 33 different items. The field test included other items that were multiple choice, but these questions were filtered out of the data for our use. The mean number of responses per question is 179 with the minimum being 128 and the maximum being 313. Most of the items belong to an item bundle - a grouping of items that are related in context and/or share a common question context. Additionally, the items were administered in four different test forms, where some items were included in multiple forms. The items all relate to four constructs about student understanding of algebra.

An example of one of the items, labelled 'Crude Oil 4ab' is included in Figure 1. For this item, students are presented with two images relating to oil production - one being a line graph and the other, a table. With the given context, students are presented with a choice between the graph or the table for which would be better to represent the historical patterns, or change over time of the oil production. The correct answer for this question is the graph, and students are expected to provide reasons why this is the right choice.

An example of a student response to the Crude Oil 4ab question (shown in Figure 1) rated at the highest (most correct)

The following graph and table show annual crude oil production in million tonnes of oil equivalent (mtoe) from 1960 to 2014.



Year	Oil Produced (mtoe)	Year	Oil Produced (mtoe)	Year	Oil Produced (mtoe)
1960	1,020	1980	2,986	2000	3,507
1961	1,087	1981	2,814	2001	3,504
1962	1,179	1982	2,695	2002	3,470
1963	1,267	1983	2,671	2003	3,595
1964	1,369	1984	2,732	2004	3,763
1965	1,467	1985	2,695	2005	3,833
1966	1,301	1986	2,807	2006	3,828
1967	1,503	1987	2,835	2007	3,816
1968	1,705	1988	2,945	2008	3,864
1969	1,904	1989	2,991	2009	3,817
1970	2,106	1990	3,026	2010	3,976
1971	2,303	1991	3,011	2011	4,006
1972	2,505	1992	3,023	2012	4,117
1973	2,619	1993	3,046	2013	4,136
1974	2,615	1994	3,111	2014	4,240
1975	2,465	1995	3,181		
1976	2,663	1996	3,254		
1977	2,747	1997	3,357		
1978	2,769	1998	3,426		
1979	2,880	1999	3,380		

[a] For a group project, you and your classmates have to present the overall historical patterns of annual oil production as a poster. Due to limited space, only one of the following representations can be included in the poster: a table, a graph, or a set of equations. Which representation should you use?

[b] Briefly explain your answer choice in [a].

Figure 1: An example of an item from the data set.

score category is shown: *"the graph easily displays patterns over time whereas the table and equations require more analyzing."* In contrast, a student response to the same question rated at the lowest (most incorrect) category is shown: *"the table is more clear, the information is seen in the table."*

Responses were rated from 0 (fully incorrect) to 4 (fully correct) by multiple researchers and subject-matter experts at the BEAR center. The quality and consistency of ratings were evaluated by an inter-rater reliability score, and when a high percentage of rating mismatches between raters existed, incongruous ratings were discussed until a consensus was reached by the raters.

4. METHODS

In this section, We briefly introduce the representation methods and model classes that we include in our experiments. Additionally, we describe the question related information, beyond that of the responses, that are used as inputs to the model. Further, we outline our experiments in detail including methods for evaluation and comparison.

4.1 Input Representations

We chose to include three distinct, yet commonly used, representation types in our experiments: A count-based method that elicits the distinct vocabulary of our data (Bag of Words), a simple neural method that utilizes pre-trained word vectors (Word2Vec), and a state-of-the-art, contextual neural method (Sentence-BERT). A short description of each representation type is included.

4.1.1 Sentence-BERT

Sentence-BERT (SBERT) is a modification of the BERT network that utilizes siamese and triplet network structures to create semantically meaningful sentence embeddings [22]. SBERT fine-tunes the BERT network on a combination of the SNLI dataset and the Multi-Genre NLI datasets, totaling about 1 million sentence pairs. Although sentence embeddings can be derived from the original BERT model using methods such as averaging the BERT output layers or using the [CLS] token embedding, it has been shown that such methods yield poor sentence embeddings [20]. In comparison, SBERT sentence embeddings outperformed other state-of-the-art methods such as InferSent [8] and Universal Sentence Encoder [5] on the SentEval [8] benchmark, which gives an idea of the quality of sentence embeddings for various tasks such.

4.1.2 Word2Vec

Word2Vec (W2V) ma13, is a neural model that creates vector representations of words that have been shown to be semantically meaningful and useful in different NLP tasks [22]. We use an extension of the previously introduced Skip-gram model [18] that incorporates sub-sampling of frequent words during training in order to speed up training, and improves accuracy of representations of less frequent words. For this project, we use the Google News corpus of pre-trained word embeddings. Vectors of size 300 are created for each word, and in order to construct response embeddings from the individual word vectors, we employ a simple but popular method: averaging the vectors of all words in the response.

4.1.3 Bag-of-words

The bag-of-words (BOW) model represents a document as a vector, or "bag," of length equal to the number of unique words in the entire corpus and values of the vector equal to the frequency with witch its corresponding word occurred in the document. In our application, the bags are student short answers and additional question information.

4.2 Input Content

In an item design context, there are various untapped sources of information relating to a particular question that may be useful to include as input to a classification model. We explore the use of four different sources of information, outside that of the response itself. A brief description of the such sources are included below.

4.2.1 Question Text

The question text consists of the direct question stem. As in the example item provided in Figure X, the question text would be: *"Briefly explain your answer choice in [a]."*

4.2.2 Question context

The question context includes any textual information beyond that of the question stem that is related to the question, and might be useful for the respondent to produce a response. In the question example in Figure 2, the question context would include: *"The following graph and table show annual crude oil production in million tonnes of oil equivalent (mtoe) from 1960 to 2014. For a group project, you and your classmates have to present the overall historical patterns of annual oil production as a poster. Due to limited space, only one of the following representations can be included in the poster: a table, a graph, or a set of equations. Which representation should you use?"* We note that not all of the included items have question context text beyond that of the question text itself. For such items, it was not possible to include question context as part of the input.

4.2.3 Rubric Text

Level	Description	Example Response
4	Student provides a fully correct positive and negative justification for ...	"The graph best illustrates the visual trend of oil production. The table or the equation won't provide an overall ..."
3	Student provides a fully correct positive justification for selected representation ...	"The table cannot show a trend in the oil production effectively. "
2	Student provides a partial/general positive justification for selected ...	"The graph helped me with more questions."
1	Student provides incorrect justification for selected representation.	"Because the graph gives us better projections."
0	Student makes no attempt to provide a justification for selected representation.	"I am not sure why."

Figure 2: An example of a scoring rubric corresponding to the item in Figure 1.

As part of the assessment cycle as previously mentioned, an important step in a measurement process is defining the outcome space for each item through a scoring guide, which is essentially a rubric. The scoring guide includes a detailed description of the reasons for which a response would be rated at a certain score, and is used as a guide for human raters. In addition, the scoring guide often includes example responses for each rating level. An example of a scoring guide for the 'Crude Oil 4ab' item in Figure 1 is included shown in Figure 2. When the rubric text is included in the input text, we include both the level descriptions as well as the example response(s).

4.2.4 *Bundle Identifier*

As described above in the data section, most of the questions belong to a bundle of questions - those that are linked based on a similar image, or context text. As a question bundle identifier, we concatenate a one-hot vector to the input vectors. Although items within the same bundle will often share the same context text, we include the one-hot bundle identifier within our extra input text experiments so that we can infer whether the model makes use of semantics within the context text, or rather just a general indication of similar questions.

4.3 Classification Models

We compare a multinomial logistic regression model with a simple neural network classification model. We chose these classification methods representing a linear transformation of the feature space to a label (regression) and a non-linear transformation (neural network). A brief overview of each model is included.

Multinomial logistic regression (MLR) is a classification model that predicts probabilities of different outcomes for a categorical dependent variable. In order to generalize to a K-class setting, the model runs K-1 independent binary logistic regression models where one outcome is chosen as a "pivot" and other K-1 outcomes are separately regressed against the pivot outcome. We use the Limited-memory Broyden-Fletcher-Goldfarb-Shannon (LBfGS) algorithm for optimization [9], and incorporate L2 regularization.

Additionally, we use a simple feed forward neural network on a categorical cross entropy loss function with 2 hidden layers of size 100, using rectified linear unit (ReLU) activation functions for both hidden layers. We include dropout of 0.4 and utilize Adam optimization. We train the model for 16 epochs and use a batch size of 36.

4.4 Evaluation and Model Comparison

In this section, we enumerate our experiments and the evaluation methods chosen for comparison.

4.4.1 *Experiments*

As input to our model, we experiment with 8 different combinations of content to vectorize and concatenate to the response vectors before training our classification models:

1) response
2) question + response
3) question context + response
4) scoring rubric + response
5) bundle one-hot + response
6) question + scoring rubric + response
7) bundle one-hot + question + scoring rubric + response
8) bundle one-hot + question + question context + scoring rubric + response

For each of the 8 combinations listed above, we create three different vector representations with the aforementioned methods: 1) Sentence-BERT, 2) Word2Vec, and 3) Bag-of-words, resulting in 24 distinct input types. We fit a classification model for each of the input types, for both of our classification models. Thus, we compare 48 separate versions of an ASAG model with three types of evaluation.

4.4.2 *Leave-one-question/bundle-out Evaluation*

In order to assess the generalizability of the ASAG model to out-of-training-sample questions for each of the 48 experiments, we average the results of N (where N is the number of questions) independent models. For each of the N models, we train the classifier on data from N-1 questions, and test on data exclusive to the left-out-of-training question. In the case of the leave-one-question-out results, it is important to note that although the model has not seen data specific to the left-out question, it has seen questions that are part of the same question bundle and are therefore related.

To expand our evaluation of generalizability further, we include a leave-one-bundle-out metric for each experiment. For such, we average the results from M (where M is the number of bundles) independent models where we train the classifier on data from M-1 bundles, and test on data exclusive to the left-out-of-training questions which belong to a single bundle. So, these results give us an idea of whether the model can successfully rate responses from questions that have not been used for training, and when the model has not seen questions related by context during training.

4.4.3 *Evaluation Metrics*

We report our results in both multilabel accuracy, and weighted F1 score because multilabel accuracy is both widely used and easy to interpret, and the weighted F1 score captures both the precision and recall and accounts for class imbalance.

Multilabel accuracy represents the degree to which our model classifications agree with the ground truth labels (for this

Table 1: Experiment Results: Multilabel Accuracy

	Response Text	Bundle ID	Question Text	Context Text	Rubric Text	Random Holdout			Question Holdout			Bundle Holdout				
						SBERT	W2V	BOW	SBERT	W2V	BOW	SBERT	W2V	BOW	Average (ACC)	Average (Wf1)
Majority Class							0.34715			0.37044			0.30521		0.34093	0.25429
LogReg	x					0.58034	0.49765	0.54665	0.35870	0.35338	0.36517	0.31806	0.25316	0.26995	0.39367	0.38323
LogReg	x	x				0.59603	0.53154	0.55602	0.37225	0.37595	0.36097	0.32855	0.25784	0.28364	0.40698	0.39513
LogReg	x		x			0.63062	0.55748	0.58702	0.40378	0.41290	0.32506	0.36276	0.30162	0.25982	0.42678	0.40317
LogReg	x			x		0.62972	0.56036	0.58486	0.40352	0.41815	0.33053	0.34539	0.31713	0.27005	0.42886	0.40411
LogReg	x				x	0.61657	0.55982	0.58846	0.38488	0.38379	0.42198	0.28175	0.23273	0.31443	0.42049	0.39944
LogReg	x		x		x	0.61818	0.56432	0.59152	0.40967	0.38968	0.37571	0.34737	0.26432	0.30380	0.42940	0.40213
LogReg	x	x	x		x	0.62484	0.56144	0.59261	0.41534	0.40174	0.36048	0.32196	0.25499	0.28627	0.42441	0.40195
LogReg	x	x	x	x	x	0.61406	0.55963	0.59009	0.41494	0.39999	0.36104	0.31797	0.26117	0.29534	0.42380	0.39920
NN	x					0.60249	0.56450	0.60973	0.35736	0.34029	0.36930	0.31540	0.25760	0.26633	0.40922	0.40709
NN	x	x				0.60540	0.59802	0.61963	0.39011	0.37083	0.35526	0.31598	0.23819	0.28716	0.42006	0.41686
NN	x		x			0.65800	0.61009	0.65532	0.40633	0.37576	0.34075	0.35574	0.32317	0.27179	0.44411	**0.42600**
NN	x			x		0.66070	0.61388	0.66198	0.39309	0.38079	0.33176	0.36227	0.31851	0.28206	**0.44500**	0.42470
NN	x				x	0.64017	0.61226	0.62234	0.36721	0.35595	0.40597	0.33015	0.23390	0.33769	0.43395	0.41403
NN	x		x		x	0.62503	0.61009	0.62198	0.41137	0.39257	0.33095	0.34568	0.25618	0.31157	0.43394	0.41456
NN	x	x	x		x	0.63206	0.59981	0.62938	0.40885	0.36204	0.39455	0.36535	0.26301	0.32980	0.44276	0.42075
NN	x	x	x	x	x	0.60557	0.59405	0.61478	0.42270	0.39130	0.35288	0.30132	0.27360	0.31397	0.43002	0.40366
Average (ACC)						**0.62124**	0.57468	0.60452	0.39500	0.38157	0.36140	0.33223	0.26919	0.29273		
Average (WF1)						**0.60830**	0.55063	0.59135	0.37680	0.35320	0.33350	0.31472	0.24976	0.28697		

project, human ratings). It is calculated simply as the number of correct predictions divided by the number of total number of examples. The F1 score for a certain class is the harmonic mean of its precision and recall, where precision is calculated as true positives divided by false positives and true positives, and recall is calculated as true positives divided by false negatives and true positives. In order to account for class imbalance, we specifically use the weighted F1 score. This metric calculates the F1 score for each class independently, and the overall score for all the classes is the average weighted by class size.

5. RESULTS

Results of our experiments are detailed in Table 1, reported in multilabel accuracy. For column and row averages, the weighted F1 score is presented as well. In the left-most half of the table, an x is present for a given row if the information type, indicated by the column header, is included in the model input. For example, results in the first row represent an input of only the response text and results in the second row represent an input of both the Bundle ID and the response. Additionally, the top half of the table results are those from the multinomial logisitic regression classifier, and the bottom half of the table results are those for the neural network classifier (as indicated by the leftmost column). For each of our evaluation methods, random holdout, question holdout, and bundle holdout, we present results for the three textual representation methods: SBERT, W2V, and BOW.

In terms of the general performance of our classification models, we consider the random holdout evaluation method. Overall, SBERT representations performed best when averaging across the classification methods and input combinations, followed by the BOW representations (accuracy of 0.621 for SBERT compared to 0.575 and 0.605 for W2V and BOW, respectively). Additionally, the neural network achieves higher accuracy than the logistic regression in general. Both SBERT and BOW perform notably well when the input includes the question text, or the question content.

To assess the generalizability to grading answers to questions unseen in the training set, we focus on the question holdout and bundle holdout results. Across the board, we see much lower accuracy for the question and bundle holdout experiments than that of the random holdout, with the bundle holdout being the lowest. This is in line with what one might expect because for the question holdout, the model has not yet seen responses for the particular question in the test set and for the bundle holdout, the model has not seen questions even related to the test set question.

Similar to the random holdout experiments, we see the same overall pattern for the question and bundle holdout experiments: SBERT is generally superior, followed by BOW and W2V, respectively. One notable difference for the question holdout experiments compared to those of the random holdout is that we see increased performance when we include multiple extra sources of information. For example, with SBERT and bundle holdout, we achieve 0.365 accuracy with the neural network classifier when we include the rubric text, question text, and bundle ID. We might explain this result as, when the model is lacking previous information about the test question from training, extra input information might provide guidance for the model.

For the question and bundle hold out experiments, the addition of the rubric text improves performances particularly well with the use of BOW representations, for both the logistic regression and neural network classifiers. With SBERT, the addition of the question text seemed to help the generalizability of the model as well. Interestingly, we do not see the same pattern between the classification models for the question and bundle holdout methods: where the neural net was clearly superior in the random holdout experiments, results are more similar between the logistic regression and neural network for the bundle and question holdout experiments.

We see from the row averages in the right most columns of the table that across all experiments and text representation types, the response and question text, as well as the response and context text achieve the highest evaluation scores. Additionally, the column averages further confirm that the SBERT representations perform best.

Further, we include results from a majority class classifier on the top row for a baseline comparison. We emphasize that, across all random holdout experiments, the classification models outperform the majority class classifier significantly. However, this is not the case for the question holdout and bundle holdout experiments. For the question holdout experiments, many of the SBERT experiments out-

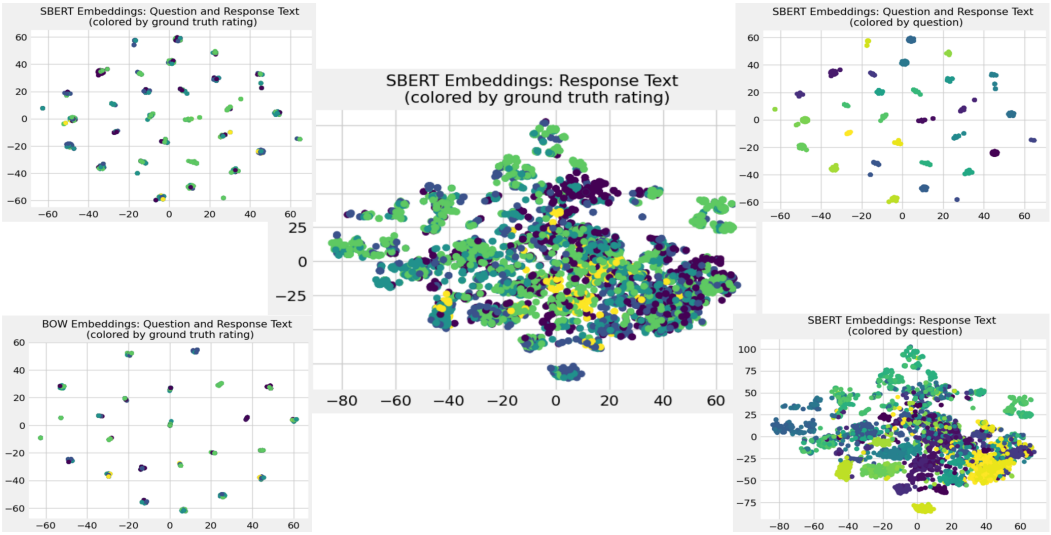


Figure 3: 2D visuals of input vector representations. Plots vary by representation type, input content, and color labeling.

performed the majority class classifier, but on average, the W2V and BOW experiments performed a bit worse than majority class. For the bundle holdout experiments, on average, SBERT performs slightly better than majority class, but the W2V and BOW experiments do not.

Further interpretations can be made from Figure 4, which includes two dimensional, t-distributed Stochastic Neighbor Embedding (TSNE) reduced vector representations. The center image includes SBERT embeddings of only the response text and the colors represent the ground truth ratings. From the top left and right images, as well as the bottom left image we can see very distinct question clusters with the inclusion of question text. However, from the bottom right image, we can visualize question specific clusters, but they are not as distinct as the representations that include the question text.

Thus, we conclude that in terms of question holdout, the model can generalize to out-of-sample questions with only slight improvement over majority class, with a state-of-the-art representation method like SBERT. Certain extra pieces of input information aid our models more than others like question text and the best performing models use the neural network classifier.

6. DISCUSSION

Although our results are not promising for the generalizability of autograding models to unseen questions, we emphasize the importance of finding more generalizable models to decrease time spent on the laborous task of creating ground-truth human ratings. Our intention is that this work will influence researchers to consider further innovative methods to increase the generalizability of ASAG models. Further, because we did find that including certain question-related text may improve model performance, it may be of use to the ASAG research community to continue to explore how extra sources of information about a question may be incorporated into an ASAG system.

As is evident in our literature review, there has been increased adoption of state-of-the-art textual representation methods such as SBERT, and transformer-based models such as BERT and XLNet, within the field of NLP in Education. Our results support that such models may achieve superior performance for certain tasks.

To build on this work further, we could consider other methods, beyond that of concatenation to the input text, to include the extra question information in our model. We could further pre-train a transformer-based model such as BERT or XLNet with the extra textual information by either tuning the existing weights or altering the existing architecture with an extra encoder layer of weights trained on our text alone. Moreover, we may focus more closely on how the classification model itself might be altered such that it might better generalize to out-of-training-sample questions, instead of only focusing on the input content.

We believe that beyond model performance, the practical utility of an ASAG system must be considered in order for educators to continue to adopt new technologies that employ advanced methods in artificial intelligence. Recent years have seen vast improvements in the field of machine learning and language processing. Embracing such technologies for applications in education may be pivotal to provide the assistance that both educators and learners need. However, we do not suggest that machine learning systems such as ASAG should be used to replace human judgements in education, especially in high stakes testing scenarios. We emphasize the ASAG systems should be used to support educators, not replace them. This project represents a continued effort to explore the ways in which we can make use of new technologies to improve learning.

7. REFERENCES

[1] N. Arthurs and A. J. Alvero. *Whose Truth is the "Ground Truth"?* College Admissions Essays and Bias in Word Vector Evaluation Methods.

[2] S. Basu, C. Jacobs, and L. Vanderwende.

Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics*, 1:391–402, 2013.

[3] S. Burrows, I. Gurevych, and B. Stein. The eras and trends of automatic short answer grading. *International Journal of Artificial Intelligence in Education*, 25(1):60–117, 2015.

[4] L. Camus and A. Filighera. Investigating transformers for automatic short answer grading. In *International Conference on Artificial Intelligence in Education*, pages 43–48. Springer, Cham, 2020.

[5] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. John, N. Constant, M. Guajardo-Céspedes, S. Yuan, C. Tar, and Y. Sung. Universal sentence encoder., 2018.

[6] G. Chen, R. Ferreira, D. Lang, and D. Gasevic. *Predictors of Student Satisfaction: A Large-Scale Study of Human-Human Online Tutorial Dialogues*. International Educational Data Mining Society, 2019.

[7] M. T. Chi, N. De Leeuw, M. H. Chiu, and C. LaVancher. Eliciting self-explanations improves understanding. *Cognitive science*, 18(3):439–477, 1994.

[8] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. preprint, 2017.

[9] R. Fletcher. *Practical Methods of Optimization(2nd ed.)*. John Wiley Sons, -0-471-91547-8, New York, 1987.

[10] S. C. Fonseca, F. D. Pereira, E. H. Oliveira, D. B. Oliveira, L. S. Carvalho, and A. I. Cristea. *Automatic subject-based contextualisation of programming assignment lists*. International Educational Data Mining Society, 2020.

[11] S. K. Gaddipati, D. Nair, and P. G. Pl"oger. Comparative evaluation of pretrained transfer learning models on automatic short answer grading. arxiv. preprint, 2020.

[12] C. L. Hancock. Enhancing mathematics learning with open-ended questions. *The Mathematics Teacher*, 88(6):496, 1995.

[13] W. J. Hou and J. H. Tsao. Automatic assessment of students' free-text answers with different levels. *International Journal on Artificial Intelligence Tools*, 20(2):327–347, 2011.

[14] R. Klein, A. Kyrilov, and M. Tokman. Automated assessment of short free-text responses in computer science using latent semantic analysis. In *G. RoBling, T. Naps, C. Spannagel (Eds.), Proceedings of the 16th annual joint conference on innovation and technology in computer science education . Darmstadt: ACM*, pages 158–162. Darmstadt: ACM, 2011.

[15] T. Liu, W. Ding, Z. Wang, J. Tang, G. Y. Huang, and Z. Liu. June). automatic short answer grading via multiway attention networks. In *International conference on artificial intelligence in education.*, pages 169–173. Springer, Cham., 2019.

[16] N. Madnani, J. Burstein, J. Sabatini, and T. O. Reilly. Automated scoring of a summary writing task designed to measure reading comprehension. In J. B. Tetreault and C. Leacock, editors, *Proceedings of the 8th workshop on innovative use of nlp for building educational applications . Atlanta*, pages 163–168. Association for Computational Linguistics, 2013.

[17] J. Mason, M. Wilson, A. E. Arneson, and D. Wihardini. *A framework for the college ready algebraic thinking assessment (CRATA)*. University of California, Berkeley Evaluation and Assessment Research Center, 2017.

[18] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality., 2013.

[19] M. Mohler and R. . Mihalcea. March). *Text-to-text semantic similarity for automatic short answer grading*, 12:567–575, 2009.

[20] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. pages 1532–1543. In Proceedings of the conference on empirical methods in natural language processing, 2014.

[21] H. Qi, Y. Wang, J. Dai, J. Li, and X. Di. Attention-based hybrid model for automatic short answer scoring. pages 385–394. SIMUtools 2019. LNICST, vol. 295, . Springer, Cham, 2019.

[22] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. preprint, 2019.

[23] C. Sung, T. Dhamecha, and N. Mukhi. Improving short answer grading using transformer-based pre-training. In *International Conference on Artificial Intelligence in Education*, pages 469–481. Springer, Cham, 2019.

[24] C. Sung, T. Dhamecha, S. Saha, M. Tengfei, V. Reddy, and A. Rishi. Pre-training bert on domain resources for short answer grading. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6071–6075, Hong Kong, CH., 2019. Association for Computational Linguistics.

[25] H. Tan, C. Wang, Q. Duan, Y. Lu, H. Zhang, and R. Li. Automatic short answer grading by encoding student responses via a graph convolutional network. *Interactive Learning Environments*, pages 1–15, 2020.

[26] K. Thaker, L. Zhang, D. He, and P. Brusilovsky. Recommending remedial readings using student knowledge state. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, pages 233–244, 2020.

[27] R. Venant and M. d'Aquin. Towards the prediction of semantic complexity based on concept graphs. In *12th International Conference on Educational Data Mining*, pages 188–197, 2019.

[28] S. A. Wind and M. E. Peterson. Sa systematic review of methods for evaluating rating quality in language assessment. *Language Testing*, 2(35):161–192, 2018.

[29] Y. Xiao, G. Zingle, Q. Jia, S. Akbar, Y. Song, M. Dong, and E. Gehringer. *Problem detection in peer assessments between subjects by effective transfer learning and active learning*. 2020.

[30] X. Yang, Y. Huang, F. Zhuang, L. Zhang, and S. Yu. Automatic chinese short answer grading with deep autoencoder. In *International Conference on Artificial*

Intelligence in Education., pages 399–404. Springer, Cham, 2018.

[31] Y. Zhang, R. Shah, and M. Chi. *Deep Learning+ Student Modeling+ Clustering: A Recipe for Effective Automatic Short Answer Grading*. International Educational Data Mining Society, 2016.

Sentiment Analysis of Student Surveys - A Case Study on Assessing the Impact of the COVID-19 Pandemic on Higher Education Teaching *

Haydée G. Jiménez,
Marco A. Casanova
Departamento de Informática
PUC-Rio
Rio de Janeiro, Brazil
{hjimenez,
casanova}@inf.puc-rio.br

Anna Carolina Finamore
COPELABS
Lusófona University
Lisboa, Portugal
anna.couto@ulusofona.pt

Gonçalo Simões
Google Research
United Kingdom
gsimoes@google.com

ABSTRACT

Sentiment Analysis is a field of Natural Language Processing which aims at classifying the author's sentiment in text. This paper first describes a sentiment analysis model for students' comments about professor performance. The model achieved impressive results for comments collected from student surveys conducted at a private university in 2019/20. Then, it applies the model to different scenarios: (i) in-person classes taught in 2019 (pre-COVID); (ii) the emergency shift to online, synchronous classes taught in the first semester of 2020 (early-COVID); and (iii) the planned online classes taught in the second semester of 2020 (late-COVID). The results show that students acknowledged the effort professors did to keep classes running during the first semester of 2020, and that the enthusiasm continued throughout the second semester. Furthermore, the results show that students evaluated professors' performance for online courses better than for in-person courses.

Keywords

sentiment analysis, BERT, online classes, in-person classes

1. INTRODUCTION

The systematic evaluation of a Higher Education Institution (HEI) provides its administration with valuable feedback about several aspects of academic life, such as the reputation of the institution and the individual performance of faculty. In fact, in some countries, it is mandatory that HEIs implement self-evaluation committees, whose members are elected by the various segments of the community and whose duties include the preparation of annual reports assessing the performance of the institution on predefined aspects.

In particular, student surveys are a first-hand source of information that help assess professor performance and course adequacy. Such surveys are typically organized as a questionnaire with *closed-ended* questions, which the student answers by choosing predefined alternatives, and *open-ended* questions, which the student answers by freely writing comments on the topic of the question. Albeit interesting and useful, the analysis of open-ended questions poses challenges, such as how to summarize the comments and how to determine the sentiment of the comments.

The primary goal of this paper is to introduce a sentiment analysis model for students' comments in the context of questionnaires designed to assess professor performance, and to evaluate the model using data from student surveys applied at Brazilian University in 2019 and 2020.

Studying this particular period of time is interesting because, in early 2020, the COVID-19 pandemic forced the Brazilian University to move all classes online, taught with the help of a videoconferencing software and a Learning Management System (LMS), and they so remained throughout 2020. This change in instructional model offers the unique opportunity to compare the in-person classes in 2019 (pre-COVID scenario), with the emergency shift to online, synchronous classes in the first semester of 2020 (early COVID scenario), and with the planned online classes in the second semester of 2020 (late-COVID scenario). Therefore, the second goal of this paper is to apply the sentiment analysis model developed to the case study data to compare the overall sentiment of the students' comments about professor performance in these different scenarios.

The results reported in this paper indicate that the sentiment analysis model developed achieves good performance in the classification of the sentiment expressed by the students' comments about professor performance. This model was separately applied to the different scenarios covered by the case study data. The results show that students acknowledged the effort professors did to keep classes running during the first semester of 2020 (early-COVID sce-

*(Does NOT produce the permission block, copyright information nor page numbering). For use with ACM_PROC_ARTICLE-SP.CLS. Supported by ACM.

Haydée Guillot Jiménez, Anna Carolina Finamore, Marco Antonio Casanova and Gonçalo Simões "Sentiment Analysis of Student Surveys - A Case Study on Assessing the Impact of the COVID-19 Pandemic on Higher Education Teaching". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 353-359. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

nario), and that the enthusiasm continued throughout the second semester of 2020 (late-COVID scenario). These conclusions are justified by the peak in positive comments observed in the first semester of 2020, as compared with the other semesters. Furthermore, the results show that students evaluated professor performance for online classes better than for in-person classes. To a large extent, these remarks are consistent, for example, with the findings of a random-sample survey, conducted in late May 2020, involving more than 1,000 US college students whose classes moved from in-person to completely online in early 2020 [13]. However, they have to be cross-checked with other surveys conducted in 2019/2020 at the Brazilian University and elsewhere.

The remainder of this paper is organized as follows. Section 2 summarizes related work. Section 3 presents the case study used in the paper. Section 4 details the model for sentiment analysis. Section 5 describes the results obtained with the case study data. Section 6 contains the conclusions and directions for future work.

2. RELATED WORK

Sentiment Analysis (SA), also known as Opinion Mining, is a field of natural language processing (NLP) where the main focus is to automatically analyze people's opinions and sentiments [11]. According to Pang and Lee [15], for most of us, the decision-making process takes into consideration "what other people think". Based on this assertion, it is easy to understand why SA is very popular in several domains, such as tourism, restaurants, movies, music, and, more recently, education.

Chaturvedi et al. [5] addressed the essential task of eliminating "real" or "neutral" comments that do not express a sentiment. The article reviewed hand-crafted and automatic models for detecting subjectivity in the literature, comparing the advantages and limitations of each approach. Ahuja et al. [1] addressed the analysis of comments from one of the most popular Twitter platforms. As the comments are not structured, they used six techniques to pre-process the comments. They then applied two techniques (TF-IDF and N-Grams) to classify comments, and concluded that the TF-IDF word level of sentiment analysis is 3-4% higher than the use of N-characteristics. Prusa et al. [17] also concentrated on Twitter data. They analysed the impact of ten filter-based feature selection techniques on the performance of four classifiers. Nazare et al. [14] analyzed about 1,000 Twitter comments using various machine learning approaches, separately or in combination, to classify the comments. Unlike other articles with traditional approaches to analyze the sentiment of short texts, Li and Qiu [10] did not consider the relationship between emotion words and modifiers, but they showed how to mitigate these problems through the sentiment structure and rules that captured the text sentiment. The results of an experiment with microblogs validated the efficacy of their approach.

Analyzing comments from sales Web sites is important to detect if users are praising or criticizing the products they consume. Bansal and Srivastava [4] used the word2vec model to convert comments into vector representations using CBOW (continuous bag of words), which were fed to a classifier. Experimental results showed that Random Forests using CBOW achieved the highest precision. Khoo and Johnkhan [9] analysed comments from the Amazon Web site, using a new general-purpose sentiment lexicon, called WKWSCI Sentiment Lexicon, and compared it with five existing lexicons. Akhtar et al. [2] used classification algorithms, like Conditional Random Filed (CRF) and Support Vector Machine (SVM), to classify comments from different Indian Web sites.

Zhou and Ye [22] reviewed journal publications between 2010-2020 in SA applied to the education domain and, among others future research directions, they pointed out: (i) the need to explore SA in the learning cross-domain; (ii) consider a combination of text mining and qualitative answers (questionnaires or interviews) to understand the psychological motivation behind learning sentiment; (iii) explore the association between sentiment, motivation, cognition, and also demographic characteristics to regulate the emotions of learners. Santos et al. [19] studied SA in online students' reviews to identify factors that influence international students' choice for a HEI. They also suggested aspects that HEI managers may have to consider to attract more international students, such as: online information about (HEI) offerings, students' comments about their experiences, international environment, courses taught in English, and support to students' accommodation or expenses. Sindhu et al. [20] proposed an aspect-oriented SA system based on Long Short-Term Memory (LSTM) models. They considered two datasets with students' comments, namely: the Sukkur IBA University and a standard SemEval-2014. They suggested that the evaluation of teaching performance would have to consider six dimensions: teaching pedagogy, behavior, knowledge, assessment, experience, and general. We previously created a tool for the analysis of student comments [8] but it was limited to a fixed, manually created dictionary, which might therefore not take into account some relevant words.

The choice of a university to enroll in is a difficult decision and, at the same time, the information available on the internet is overwhelming. To address these issues, Balachandran and Kirupananda [3] proposed an aspect-based sentiment analysis tool to evaluate the reputation of universities in Sri Lanka from users' comments in Facebook and Twitter, using the StanfordCoreNLP library to perform sentiment analysis. Lytras et al. [12] built the Learning Analytics Dashboard for E-Learning (LADEL) tool to monitor different sources, such as student blogs, social networks and Massive Open Online Courses (MOOC) in search of comments that express satisfaction, anxiety, efficiency, frustration, abandonment. LADEL is composed of four modules: collection, cleaning, word cloud and sentiment of opinion. Sivakumar and Reddy [21] extracted students' comments using the Twitter API and tried to analyze the relations between word aspects and phrases of student opinion. They used a sentiment package available in R to find the polarity of the sentences and then applied k-mean clustering and naïve Bayes for the sentiment analysis classification.

de Oliveira and de Campos Merschmann [6] analyzed the combination of NLP pre-processing tasks (tokenization, POS tagging, stemming, among others) with three classifiers (Ran-

dom Forest, Support Vector Machine, and Multilayer Perceptron), and discussed their predictive performance. They evaluated these tasks in five Portuguese datasets related to sentiment analysis, encompassing comments, news and tweets. They analyzed some combinations of preprocessing tasks and classifier

This paper focuses on identifying students' sentiments expressed in comments about professor performance in Higher Education. It uses the pre-trained model called Bidirectional Encoder Representations from Transformers (BERT) [7] for the sentiment analysis task. BERT-style models are the current state-of-the-art in several NLP tasks, including entity recognition and sentiment analysis. BERT's architecture is based on multi-layered transformers, which are particularly optimized to be trained on GPUs and TPUs with significant amounts of data. For this reason, a recipe for success with these models is to pre-train them with large datasets (in the order of millions of documents) on general tasks such as masked language models or next sentence predictions [7]. This pre-training allows the model to learn a lot about some language patterns (that are independent of the task we care about) and make it easier to train them specifically for other language tasks even without the need for large amounts of annotated data. Our corresponding code is available at GitHub[1].

3. CASE STUDY

3.1 Course Survey Data

In the rest of this paper, we use *course* to denote "a series of lectures in a particular subject", and *class* to describe "a particular instance of a course". Therefore, students enroll in a class of a course. We assume that classes run on a per semester basis, and use <year>.1 and <year>.2 to denote the first and second semesters of the calendar year, respectively.

Since 2005, at the Brazilian University used in the case study, students are invited, at the end of each semester, to answer a questionnaire for each class they took in the semester. Students' participation in the survey is not mandatory. The questionnaire has a set of closed-ended questions about the professor that taught the class, and a separate set of closed-ended questions about the course the class is an instance of. For each closed-ended question, the student chooses a score from a Likert scale (1-5). The questionnaire also has one open-ended question which invites students to write as many sentences as they like to express their evaluation of the professor that conducted the class, and likewise for the course the class is an instance of. The comments are in Portuguese and the sentences are often ungrammatical. We are interested in the sentiment analysis of the students' comments about the professor performance, which we will refer to as the *comments* for brevity.

The purpose of the case study is to analyse comments collected from the questionnaires applied in the first and second semesters of 2019 and 2020. However, we also use the comments collected from the questionnaires applied in both semesters of 2018 for pre-training (see Section 5). The reason for using comments from 2018 for pre-training is that we wanted to make sure that no comment used in the analysis step has been observed before in the pre-training step. Using the 2018 data is possible because it has been observed that the vocabulary students use to write comments has not changed significantly over the years. Table 1 presents the number of comments for the 2018, 2019 and 2020 student surveys.

[1]https://github.com/hguillot/Sentiment-Analysis-of-Student-Surveys-with-BERT

Table 1: Number of comments about professor performance in classes.

Semester	#Comments
2018.1 and 2018.2	10,077
2019.1	3,182
2019.2	1,910
2020.1	3,492
2020.2	2,219

Table 2: Structure of the professor questionnaires.

Year	Class Mode	#Closed-ended Questions	#Open-ended Questions
2018	in-person	10	1
2019	in-person	16	1
2020	online	20	1

As far as professor evaluation is concerned, the questionnaires varied slightly from 2018 to 2019. Also, in early 2020, the COVID pandemic forced the university to move all classes online, taught with the help of a videoconferencing software and a Learning Management System (LMS), and they so remained throughout 2020. The questionnaire, used for classes taught in 2020.1 and 2020.2, was then modified accordingly. Table 2 summarizes the structure of the various questionnaires that the case study is concerned with.

Given this new reality, forced by the COVID pandemic, it is reasonable to ask if the professors were prepared for online classes and if this would affect the students' evaluation of the professor performance at the end of 2020.1 (the early-COVID scenario).

As a simple answer to this conjecture, consider the last closed-ended question incorporated in the 2019/20 surveys: *"O: Overall evaluation of the professor"*. Figure 1 depicts the distribution of the scores of Question O per semester, grouped as 1 and 2, for "negative", 3, for "neutral", and 4 and 5, for "positive", considering only questionnaires with a non-empty comment about professor performance. Figure 1 shows that students in fact evaluated the overall professor performance better in 2020.1 (again, the early-COVID scenario) than in the other semesters.

But the question remains if the overall sentiment of the comments about professor performance points in the same direction.

3.2 Use of the Course Survey Data

This section describes how the course survey data were used to construct models for the sentiment analysis of comments about professors performance (recall that each questionnaire

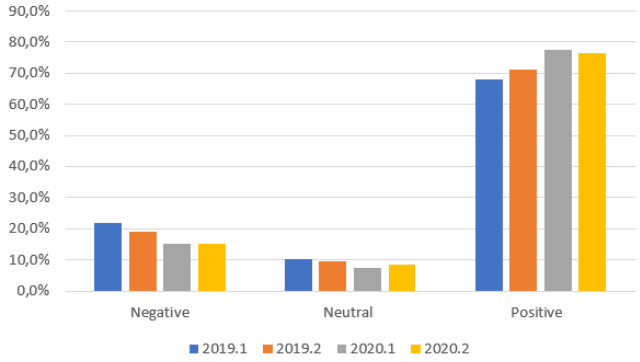


Figure 1: Distribution of the scores of Question O per semester (considering only questionnaires with a non-empty comment about professor performance).

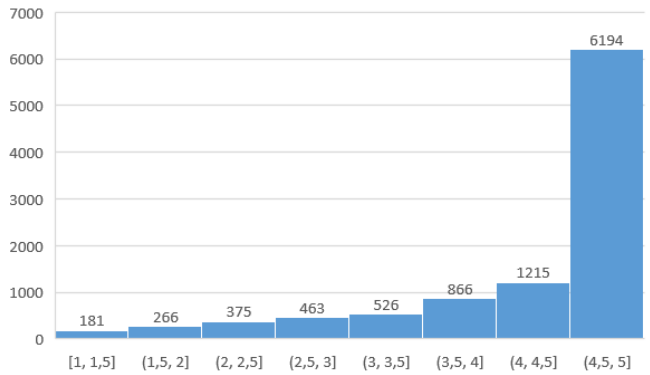


Figure 2: Distribution of the average score of all questions of a questionnaire from 2018.

has only one such comment).

We first observe that no text pre-processing was necessary, as in the Twitter sentiment analysis reported in [16], since the students' comments do not significantly depart from written Portuguese, albeit they often contain ungrammatical sentences.

The models used manually annotated comments, obtained as follows. From the course surveys of the two semesters of 2019, 800 questionnaires with non-empty professor comments were randomly chosen, using the following criteria: 5 samples were chosen for each of the Likert scale scores (1-5) for each of the 16 closed-ended questions (5 * 5 * 16 = 400) in each of the semesters (400 * 2 = 800). The comments of the selected questionnaires were manually classified into 3 categories: *positive*, when the comment only praised the professor; *negative*, when the comment only criticized the professor; and *neutral* when the comment expressed no opinion or when the comment both praised and criticized the professor. Table 3 shows the number of comments in each of these classes.

The pre-training step (see Section 5) used data from the 2018 student surveys as follows. We considered a dataset with all questionnaires with non-empty comments from the 2018 student surveys. But, since the questionnaire applied in 2018 had no overall professor evaluation (Question O), we used the average score $s_{avg}[q] \in [1, 5]$ of all questions of a questionnaire q to induce a label $c[q] \in \{$"*negative*", "*neutral*", "*positive*"$\}$ for the comment as follows: if $s_{avg}[q] < 3$ then $c[q] =$ "*negative*"; if $3 \leq s_{avg}[q] < 4$ then $c[q] =$ "*neutral*"; and if $s_{avg}[q] \geq 4$ then $c[q] =$ "*positive*". Figure 2 shows the distribution of the average scores obtained.

4. A SENTIMENT ANALYSIS MODEL

In the paper, we focus on the *polarity classification* task, whose focus is to classify comments, which express opinions or reviews, into "positive", "negative" or "neutral", or even into more than these three classes. We neither consider *subjectivity classification*, i.e., the task of verifying the subjectivity and objectivity of a comment, nor *irony detection*, i.e., the task of verifying whether the comment is ironic or not.

Table 3: Distribution of the number of questionnaires per class of comment about professor performance, using the manual classification and the automatic classification induced by the score of Question O (considering 800 questionnaires with a manually classified comment about professor performance).

Year	Classification	Positive	Negative	Neutral
2019.1	Manual	107	220	73
	Automatic	187	150	63
2019.2	Manual	119	203	78
	Automatic	201	138	61

We use BERT [7], which achieves outstanding results on a number of NLP tasks. The core of the architecture has been pre-trained on a very large amount of unlabeled data. The model is then fine-tuned on small supervised datasets, designed for the task in question.

For our case study, BERT encodes each comment into a 768-dimensional embedding and, then, a dense layer transforms the embeddings into a three-dimensional vector for each comment that indicates the probability that the comment belongs to each of the three classes - "positive", "negative" or "neutral". We adopted the BERT-Base, Multilingual Cased version[2] (for 104 languages, with 12-layer, 768-hidden, 12-heads, 110M parameters), which is required since the comments are written in Portuguese. In order to significantly speed up the training and inference with our model, we limited the size of each input comment to 64 tokens, which is enough to cover the vast majority of the comments. Any comment with less than 64 tokens was padded with the '[PAD]' symbol already allocated in BERT's vocabulary and any comment with more than 64 tokens was truncated.

Finally, the model was implemented using KERAS and running on GPU's.

5. EXPERIMENTS AND RESULTS

We started our experimental setup by executing a pre-training step that aims at getting the model used to the style of stu-

[2]Available at `https://github.com/google-research/bert/blob/master/multilingual.md`

Table 4: Results of the experiments.

Experiment	Accuracy	Precision	Recall	F1
Zero-shot	50.2±2.3	54.2±2.2	51.8±2.8	53.0±2.4
From scratch	86.3±1.8	**84.5±2.3**	83.0±3.1	83.7±2.4
Fine-tuned	**87.5±2.0**	84.6±2.0	**84.8±2.0**	**84.6±2.5**

dent's comments through non-annotated data. In order to do that, we used the set of comments from the 2018 student surveys, and the scores they assigned to the professor as a proxy to the labels, as explained in Section 3.2. We started the pre-training experiment with the multilingual BERT checkpoint that is publicly available and trained for 10 epochs, resulting in a newly trained checkpoint which we call from this point on the *pre-trained checkpoint*.

After the pre-training step, we proceeded to experiment with three setups, using a 5-fold cross-validation strategy, applied to the set of 800 manually classified comments. Therefore, each round of cross-validation used 640 comments for training and 160 comments for testing. The three setups we used were as follows:

- *Zero-shot*: this experiment does not perform any training with the manually classified comments. Instead, it performs inference directly using the pre-trained checkpoint that resulted from the pre-training step on the test set. If this model's performance was good, then it would show that manually annotating comments would not be necessary.
- *From scratch*: this experiment does not use the pre-trained checkpoint that resulted from the pre-training step. Instead, it starts with the multilingual BERT checkpoint and uses the manually classified comments to train and evaluate the model. The objective of this experiment is to understand if the pre-training step is necessary to obtain top-quality results.
- *Fine-tuned*: this experiment uses the pre-trained checkpoint that resulted from the pre-training step and then uses it as the starting point when training with the manually classified documents. This experiment aims at evaluating if combining pre-training and manually annotated comments helps in obtaining top-quality results.

Table 4 shows the results of the 5-fold cross-validation (each cell indicates the average and the standard deviation over the 5 rounds). Observe that the fine-tuned model obtained the best results, which indicates that combining pre-training and manually annotated comments helps in obtaining top-quality results.

We have also computed the Fisher-Irwin test [18], to examine the hypothesis that Fine-tuned model does not have an equivalent classification performance when compared to both Zero-shot and From scratch. For this purpose, we computed the Fisher-Irwin test twice. In the first test, our null hypothesis (*Fine-tuned classifier has a proportion of correct classifications equivalent to the proportion of correct classifications from Zero-shot classifier*) was tested against the alternative hypothesis (*Fine-tuned classifier has a proportion of correct classifications superior to the proportion of correct classifications from the Zero-shot classifier*), and the null hypothesis was rejected for the usual levels of statistical significance (5% and 10%). The same happened in our second test where our null hypothesis (*Fine-tuned classifier has a proportion of correct classifications equivalent to the proportion of correct classifications from From scratch classifier*) was tested against the alternative hypothesis (*Fine-tuned classifier has a proportion of correct classifications superior to the proportion of correct classifications from the From scratch classifier*). Based on this, we can conclude that our results are statistically significant, since our null hypotheses were both rejected for the usual levels of statistical significance (5% and 10%), leading us to accept alternative hypotheses.

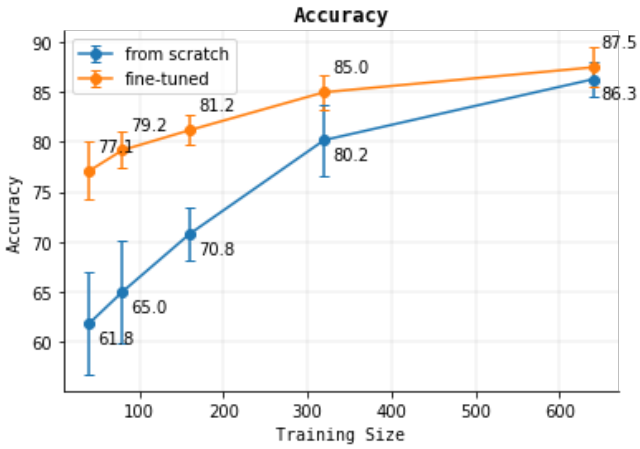


Figure 3: Accuracy for *From scratch* and *Fine-tuned* using train set of 40, 80, 160, 320 and 640 comments.

An important question that arises is about the number of comments that must be manually annotated to achieve an acceptable level of accuracy. To address this question, we ran the following cross validation experiment, with a decreasing number of manually annotated comments used for training. We divided the 800 manually annotated comments into 5 sets of 160 comments each. Let $G_1, ..., G_5$ denote these sets and $\overline{G_i}$ denote the 640 comments not in G_i. For each $i = 1, ..., 5$, we computed the accuracy and the F1-score of the from-scratch and the fine-tuned models, using G_i for testing and subsets of $\overline{G_i}$, of sizes 640, 320, 160, 80, and 40, for training. Finally, for each cardinality of the training sets, we computed the average accuracy and the average F1-score of each model. Figures 3 and 4 depict the results.

Figure 3 shows that, using 640 manually annotated comments for training, the fine-tuned model achieved an average accuracy of 87.5% and the from-scratch model achieved 86.3%, and so on for the other training set cardinalities (320, 160, 80 and 40). Therefore, based on the level of accepted accuracy, one can balance the effort to manually annotate the comments.

Figure 3 also shows that: (i) using just 40 manually annotated comments for training, the fine-tuned model achieved an average accuracy of 77.1%, while the from-scratch model

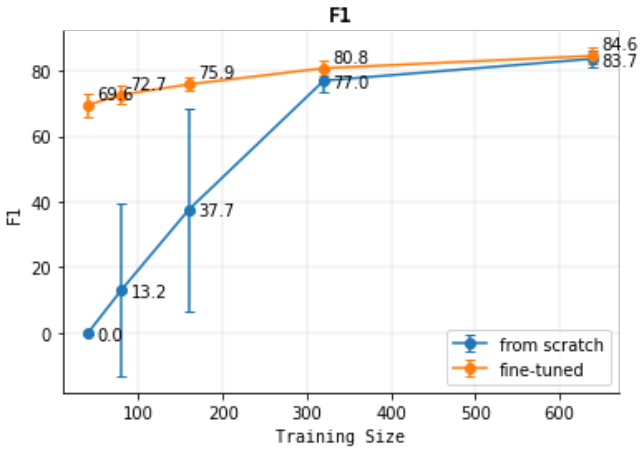


Figure 4: F1 for *From scratch* and *Fine-tuned* using train set of 40, 80, 160, 320 and 640 comments.

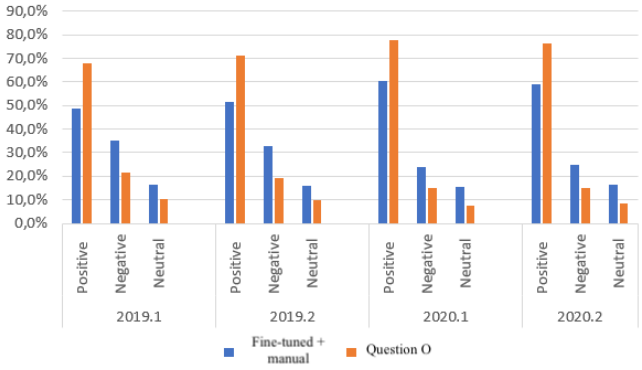


Figure 5: Distribution of the final classification of the comments from all surveys, using the fine-tuned model, added to the manually classified comments from 2019.1 and 2019.2 (shown in blue), and the classification of the comments from all surveys, using the score of Question O (shown in orange).

only achieved an accuracy of 70.8%, when trained with 160 comments, that is, 4 times as much comments; (ii) the fine-tuned model, again trained with just 40 comments, achieved a much better accuracy than that of the zero-shot model, shown in the first line of Table 4 (the zero-shot model is the equivalent to training the fine-tuned model with 0 comments); (iii) the pre-trained check-point had a positive impact, since the fine-tuned curve is always above the from-scratch curve; (iv) the fine-tuned model achieved a standard deviation smaller than that of the from-scratch model, which means that this technique is more stable and less susceptible to changes due to the samples. These observations reinforce that, with an adequate pre-training strategy, we may achieve good results without the need to manually annotate a large amount of data.

Finally, we used the fine-tuned model to classify the full set of comments from the 2020.1 and 2020.2 surveys, and the set of comments from 2019.1 and 2019.2 that were not manually classified. Then, we added the manually classified comments from 2019.1 and 2019.2 to obtain the final distributions for the four semesters, as shown in Figure 5.

For comparison purposes, Figure 5 includes the distributions of the comment classifications induced by the score of Question O as explained in Section 3.1. Note that Question O induces a classification biased towards positive comments, when compared with the classification based on the fine-tuned model. This is also observed when just the manually classified comments are considered.

In conclusion, the distributions of the students' comments sentiment and of the scores of Question O indicate that students evaluated the professor performance better in 2020.1 (the early-COVID scenario) than in the other semesters, which seems to indicate that students acknowledged the effort professors did to keep classes running during 2020.1, and that the enthusiasm continued throughout 2020.2 (late-COVID scenario). Furthermore, students evaluated the professor performance better in 2020.1 and 2020.2 (online classes), by a margin of nearly 10%, when compared with 2019.1 and 2019.2 (in-person classes), respectively.

6. CONCLUSIONS

This paper first described a sentiment analysis model for students' comments about professor performance. The model is based on BERT and has achieved good results when applied to a case study with students' comments about professor performance, obtained in 2019/20.

Then, the paper applied the model to compare the overall sentiment of the students' comments about professor performance in different scenarios: in-person classes in 2019.1 and 2019.2 (pre-COVID scenarios); the emergency shift to online, synchronous classes in 2020.1 (early COVID scenario); and the planned online classes in 2020.2 (late-COVID scenario). The results show that students acknowledged the effort professors did to keep classes running during 2020.1, and that the enthusiasm continued throughout 2020.2. Furthermore, the results show that students evaluated professor performance for online courses better than for in-person courses, by a margin of nearly 10%, which seems to indicate that students favor online classes.

This paper also discussed the number of comments that must be manually annotated to achieve good results. Future experiments can take advantage of this discussion to reduce the manual annotation effort, even with datasets obtained from other universities.

The stability of the models was also investigated, indicating that the fine-tuned model achieved a lower standard deviation, which means that this technique leads to more stable results. The fine-tuned model also achieved a higher performance, when compared to both the zero-shot and from-scratch models, in terms of the proportion of correct classifications, and the difference was statistically significant.

We plan to extend the analysis to past student surveys, which go back to 2005, and to the student survey to be applied at the end of 2021.1, when classes will still be online. We also plan to cross-check these preliminary findings with other surveys conducted in 2019/20 at the Brazilian University and elsewhere.

7. ACKNOWLEDGMENTS

This work was partly funded by FAPERJ under grant E-26/202.818/2017, by CAPES under grant 88882.164913/2010-01, and by CNPq under grants 302303/2017-0 and 162959/2017-6.

8. REFERENCES

[1] R. Ahuja, A. Chug, S. Kohli, S. Gupta, and P. Ahuja. The impact of features extraction on the sentiment analysis. *Procedia Computer Science*, 152:341–348, 2019.

[2] M. S. Akhtar, A. Ekbal, and P. Bhattacharyya. Aspect based sentiment analysis in Hindi: Resource creation and evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2703–2709. European Language Resources Association (ELRA), May 2016.

[3] L. Balachandran and A. Kirupananda. Online reviews evaluation system for higher education institution: An aspect based sentiment analysis tool. *2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pages 1–7, 2017.

[4] B. Bansal and S. Srivastava. Sentiment classification of online consumer reviews using word vector representations. *Procedia Computer Science*, 132:1147–1153, 2018. International Conference on Computational Intelligence and Data Science.

[5] I. Chaturvedia, E. Cambria, R. E. Welsch, and F. Herrera. Distinguishing between facts and opinions for sentiment analysis: Survey and challenges. *Information Fusion*, 44:65–77, December 2018.

[6] D. N. de Oliveira and L. H. de Campos Merschmann. Joint evaluation of preprocessing tasks with classifiers for sentiment analysis in brazilian portuguese language. *Multimedia Tools and Applications*, pages 1–22, 2021.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] H. G. Jiménez, M. A. Casanova, B. P. Nunes, and A. C. Finamore. Courseobservatory: Sentiment analysis of comments in course surveys. In M. Chang, D. G. Sampson, R. Huang, A. S. Gomes, N. Chen, I. I. Bittencourt, Kinshuk, D. Dermeval, and I. M. Bittencourt, editors, *19th IEEE International Conference on Advanced Learning Technologies, ICALT 2019, Maceió, Brazil, July 15-18, 2019*, pages 176–178. IEEE, 2019.

[9] C. S. Khoo and S. B. Johnkhan. Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons. *Journal of Information Science*, 44(4):491–511, 2018.

[10] J. Li and L. Qiu. A sentiment analysis method of short texts in microblog. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 1, pages 776–779, 2017.

[11] B. Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[12] M. D. Lytras, E. D'Avanzo, P. Adinolfi, I. Novo-Corti, and J. Picatoste. Sentiment analysis to evaluate teaching performance. *Int. J. Knowl. Soc. Res.*, 7(4):86–107, October 2016.

[13] B. Means and J. Neisler. Suddenly online: A national survey of undergraduates during the covid-19 pandemic, 2020.

[14] S. P. Nazare, P. S. Nar, A. S. Phate, and P. D. D. R. Ingle. Sentiment analysis in twitter. *International Research Journal of Engineering and Technology (IRJET)*, 5(1):880–886, January 2018.

[15] B. Pang and L. Lee. Opinion mining and sentiment analysis (foundations and trends (r) in information retrieval), 2008.

[16] M. Pota, M. Ventura, R. Catelli, and M. Esposito. An effective bert-based pipeline for twitter sentiment analysis: A case study in italian. *Sensors*, 21(1), 2021.

[17] J. D. Prusa, T. Khoshgoftaar, and D. Dittman. Impact of feature selection techniques for tweet sentiment classification. In *FLAIRS Conference*, 2015.

[18] S. M. Ross. *Introduction to probability and statistics for engineers and scientists*. Academic Press, 2020.

[19] C. L. Santos, P. Rita, and J. Guerreiro. Improving international attractiveness of higher education institutions based on text mining and sentiment analysis. *International Journal of Educational Management*, 2018.

[20] I. Sindhu, S. M. Daudpota, K. Badar, M. Bakhtyar, J. Baber, and M. Nurunnabi. Aspect-based opinion mining on student's feedback for faculty teaching performance evaluation. *IEEE Access*, 7:108729–108741, 2019.

[21] M. Sivakumar and U. S. Reddy. Aspect based sentiment analysis of students opinion using machine learning techniques. In *2017 International Conference on Inventive Computing and Informatics (ICICI)*, pages 726–731. IEEE, 2017.

[22] J. Zhou and J.-m. Ye. Sentiment analysis in education research: a review of journal publications. *Interactive Learning Environments*, pages 1–13, 2020.

Context-aware Knowledge Tracing Integrated with The Exercise Representation and Association in Mathematics

Tao Huang[1], Mengyi Liang[2], Huali Yang[1], Zhi Li[2], Tao Yu[2] and Shengze Hu[1]
[1] National Engineering Laboratory for Educational Big Data, Central China Normal University, Wuhan, China
{tmht, yanghuali}@mail.ccnu.edu.cn, shengzehu@mails.ccnu.edu.cn
[2] National Engineering Research Center for E-Learning, Central China Normal University, Wuhan, China
{mengyiliang, zhili, yt2542106000}@mails.ccnu.edu.cn

ABSTRACT
Influenced by Covid-19, online learning has become one of the most important forms of education in the world. In the era of intelligent education, knowledge tracing(KT) can provide excellent technical support for individualized teaching. For online learning, we come up with a new knowledge tracing method that integrates mathematical exercise representation and association of exercise(ERAKT). In the aspect of exercise representation, we represent the multi-dimensional features of the exercises, such as formula, text and associated concept, by using ontology replacement method, language model and embedding technology, so we can obtain the unified internal representation of exercise. Besides, we utilize the bidirectional long short memory neural network to acquire the association between exercises, so as to predict his performance in future exercise. Extensive experiments on a real dataset clearly proved the effectiveness of ERAKT method, they also verified that adding multi-dimensional features and exercise association can indeed improve the accuracy of prediction.

Keywords
Knowledge tracing. Context-aware. Exercise representation.

1. INTRODUCTION
As one of the key technologies of adaptive learning, knowledge tracking has become a research hotspot in adaptive education. The main task of knowledge tracking is to automatically track students' acquisition knowledge level with time according to their historical learning trajectory, so as to accurately predict their performance in future learning. In actual teaching, teachers can adjust teaching plan dynamically by predicting the result, improve teaching quality and teaching efficiency, and help teachers to achieve accurate teaching goal.

Knowledge Tracing method (KT) was first proposed by Atkinson. Bayesian knowledge tracing method (BKT) [1] is one of the most popular knowledge tracing methods in the early stage. BKT assumes that students will never forget a knowledge concept once they have mastered it, which is not in line with the actual teaching situation. Later, with the continuous development of deep learning, more and more scholars combined knowledge tracing tasks with deep learning methods, among which Deep Knowledge Tracing (DKT) [2] is the most popular and commonly used one. DKT partly solves the assumption error problem in BKT which does not conform to the actual teaching situation, and can more accurately represent the concept proficiency of learners. However, the assumption of concept state represented by a hidden layer in DKT is inaccurate, making a student's mastery level difficult to track. Furthermore, Jiani Zhang et al. proposed Dynamic Key-Value Memory Networks for Knowledge Tracing (DKVMN) [3] based on memory neural networks, and DKVMN is significantly better than BKT and DKT in terms of performance effect. In recent years, the University of Science and Technology of China team proposed some methods which integrated exercise records and exercise materials into KT based on the existing KT methods, such as EKT[11], qDKT[12], etc., which has stronger explanatory power and gradually improved performance effect.

However, most of the traditional knowledge tracing methods only consider the exercises records of students, using the covered concepts to index the exercises, ignoring the influence of exercise formula, text or concepts on a student's knowledge state. In fact, besides exercise interaction records, the multi-dimensional information of exercises has an important impact on a student's performance. Therefore, in order to solve the above problems, we propose a mathematical knowledge tracing method that integrates the representation and association of a student's exercises, so as to solve the problem of information loss caused by ignoring multi-dimensional representation and association of exercises in traditional knowledge tracing and improve accuracy of the method. Contributions of ERAKT are as follows:

(1) We propose a new context-aware knowledge tracing method that can automatically learn and predict a student's performance in the next exercise.
(2) We propose an exercise representation method that integrates multi-dimensional information, including text, formulate and associated concept.
(3) We propose a sequential question association mining method based on a bidirectional neural network to acquire association content between exercises.

2. RELATED WORK
2.1 Semantic representation
In the domain of text processing, the most important task is to transform text into a vector form which could be understood and processed by computers, that is, semantic representation. There are

Tao Huang, Mengyi Liang, Huali Yang, Zhi Li, Tao Yu and Shengze Hu "Context-aware Knowledge Tracing Integrated with The Exercise Representation and Association in Mathematics". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 360-366. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

many ways to get semantic representation, such as Word2vec, TextCNN, FastText, Bert etc.

Word2Vec[4] was proposed by Google's Mikolov team. It is the preliminary application of neural networks in semantic representation. Word2Vec conducts fixed-dimensional vectors to represent words. For sentences, the Doc2Vec method [5] is derived, which establishes a model by means of a neural network structure. Paragraph vectors are obtained during the training of the model. Both of them are typical unsupervised text representation methods. Compared with the traditional bag-of-words method, they can better integrate the internal information of exercise, such as context, semantics and word order.

Yoon Kim modified the input layer of traditional CNN. In 2014, he proposed a text classification method named TextCNN[6], which has a simpler network structure, smaller amount of calculations and faster speed of training. So compared with traditional CNN method, TextCNN performs better in field of semantic representation. Another method to get semantic representation is FastText [7]. FastText can train word vectors by itself without requiring pre-trained word vectors, which speeding up training and testing while maintaining high accuracy.

The Bert [8] method was also proposed by the Google team to solve the semantic representation problem. To deal with the effects of polysemous words in sentences, Bert exploits the transfomer model with Self-attention and Multi-Headed Attention mechanisms [9], which combines the context of the sentence to determine the specific semantics. Bert has a two-way function, allowing for more accurate results and adaptive learning in a multi-tasking environment.

2.2 Knowledge tracing considering multi-dimensional characteristics of exercises

With the rapid development of deep learning, more and more scholars exploit different deep learning methods to represent exercises in order to complete the knowledge tracking task and attempt to comprehensively consider the impact of different features of the exercises on knowledge tracing tasks. The multi-dimensionality mainly include textual materials and concepts involved in the exercises.

Therefore, the majority of existing KT methods utilize concepts to index exercises to avoid over-parameterization. For example, both DKT and DKVMN treat all the exercises covering the same concept as a single one. Compared with the former, the key-value matrix in DKVMN extends the hidden feature representation of the exercises, but it still does not take advantage of the characteristics of other dimensions. The Prerequisite-driven deep knowledge tracing(PDKT) [23] method integrates the structural information between concepts with the help of the Q matrix in the cognitive diagnosis theory, and specifically considers the contextual relationship between concepts. The self-attentive knowledge tracing (SAKT) [24] method exploits concepts to index exercises and introduces a self-attention mechanism to consider the degree of relevance between concepts. The Context-Aware Attentive Knowledge Tracing method(AKT) [13] utilizes a novel monotonic attention mechanism that relates a student's future responses to assessment exercises to their past responses; attention weights are computed using exponential decay and a context-aware relative distance measure, in addition to the similarity between exercises. Moreover, AKT utilizes the Rasch model to capture individual differences between exercises. The Graph-based Knowledge Tracing(GKT) [25] also introduces concepts to index exercises, at the same time constructs a graph method to represent the association between concepts, updates the student's knowledge status through the GRU mechanism.

The exercise materials which KT methods consider are mainly text materials and the concepts covered. EERNN (Exercise-Enhanced Recurrent Neural Network) [10] and qDKT(Question centric Deep Knowledge Tracing)[12] predict a student's performance only by making full use of his practice records and text of exercises. EKT (Exercise-aware Knowledge Tracing for Student Performance Prediction) [11] is an improved method based on EERNN. It is the first method to comprehensively consider the influence of a student's practice records and exercise materials (concepts and text contained) on his performance. But it is worth noticing that in EKT exercise text is represented by LSTM, due to its internal structure problem, LSTM can't parallel computing, resulting in dealing with text slower, so the effect is not very satisfactory. Exploring Hierarchical Structures for Recommender Systems (EHFKT)[21] make full use of concepts, difficulty, and semantic features to represent exercises. The first two are embedded using TextCNN, and semantic features are extracted using Bert. The Introducing Problem Schema with Hierarchical Exercise Graph for Knowledge Tracing(HGKT) [22] method exploits a hierarchical graph neural network to learn the graphical structure of the exercises. It also introduces two attention mechanisms to better mine knowledge state of learners, and utilizes the K&S diagnosis matrix to obtain the diagnosis result.

3. PROPOSED METHOD

The student's knowledge mastery is tracked by observing his interactive information in different exercises. It is a supervised learning sequence prediction problem in the field of machine learning. In this section, we will first define the problem and then describe the proposed method in detail.

3.1 Problem Definition

Assuming that each student does exercises separately, we define the student sequence $s = \{(e_1, r_1), (e_2, r_2), \dots\dots, (e_T, r_T)\}$, where e_T belongs to exercises sequence E, which represents the exercises done by the student at time T. Usually, 0 or 1 is applied to mark whether his answer is correct, i.e. 1 means correct, 0 means wrong. In the ERAKT method, we add the text content of each exercise into the student's exercise sequence E, because we mainly focus on knowledge tracing task in mathematics, which generally contain not only words but also specific mathematical elements, so the text is represented as $e = \{w, f\}$, where $w = \{w_1, w_2, \dots\dots, w_M\}$ represents the words in the exercise, and $f = \{f_1, f_2, \dots\dots, f_M\}$ represents the mathematics components. Furthermore, the corresponding concept is a key information in KT task, $k = \{k_1, k_2, \dots\dots, k_M\}$, which is summarized into a concepts matrix for embedding, and the student sequence after embedding turns into sequence $s = \{(e_1, k_1, r_1), (e_2, k_2, r_2), \dots\dots, (e_T, k_T, r_T)\}, s \in S$.

The ultimate goal of our ERAKT method is to track a student's knowledge status through hidden layers to predict his performance in future exercise, that is, his response to exercise e_{T+1} at the next moment T+1. Besides, we take into account the student's record of exercises, the text content of the exercises and the concepts included. The ERAKT framework is shown in Figure 1. The method includes three major parts: exercise representation module, exercise association module and the performance prediction module.

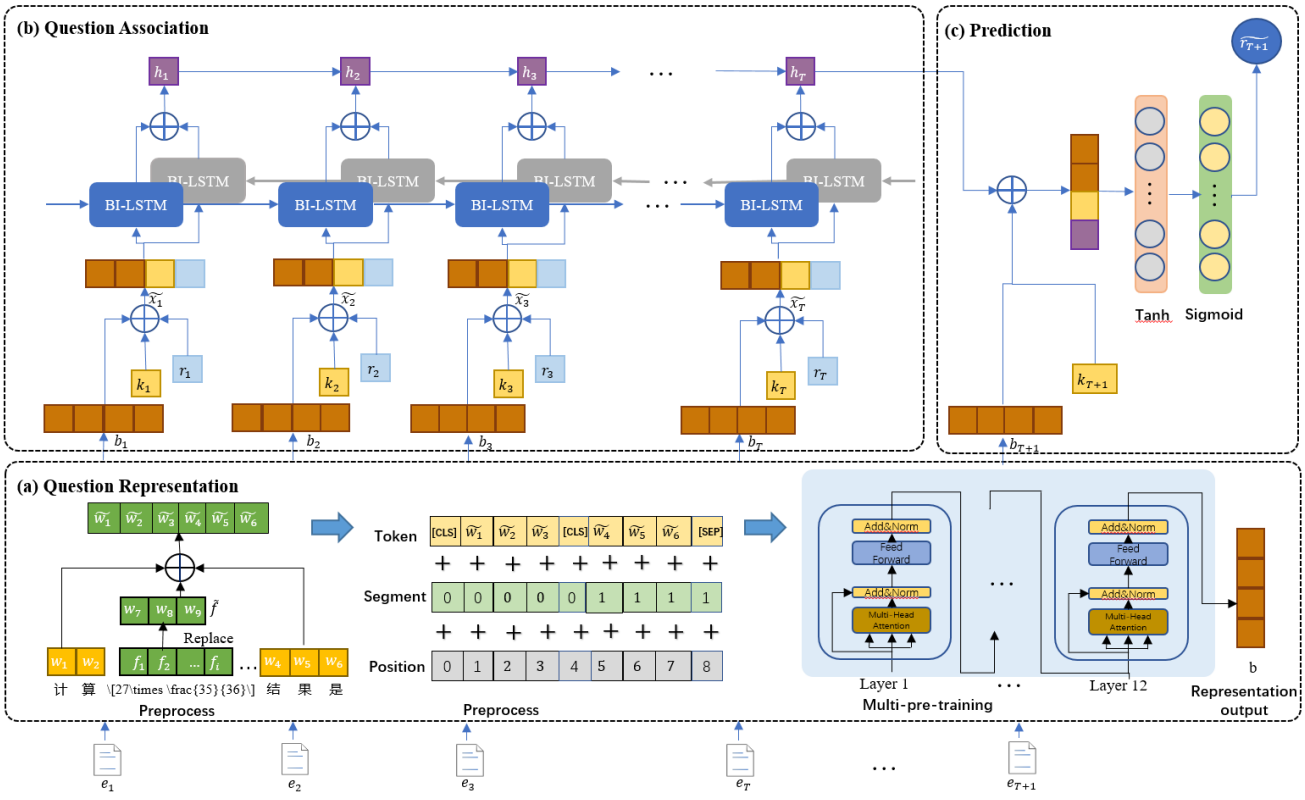


Figure 1. ERAKT framework. (a) The process of obtaining the exercise representation. After the replacement of the formula, we splice the formula with the original text, and the final exercise representation vector is obtained through 3 embedding layers and 12 layers of encoder. (b) Combine the exercise representation vector with the knowledge concepts and student response corresponding to the exercise, and input them into the Bi-LSTM network together to obtain the student's hidden state representation. (c) The student response prediction part, which predicts the student's response to the exercise at time T+1.

3.2 Exercise Representation Method

The information of the exercise includes the material itself and the concepts associated with it. To realize the unified representation of mathematics exercises, firstly, we need to construct the different dimension features in the exercises to represent the material itself and its associated concepts, and then integrate the feature representations of multidimensional exercises into a unified feature vector.

3.2.1 Preprocessing of exercise formula

Mathematical exercises involve text and formulas, which are represented by LaTex, and we need to convert formulas into unified text expressions in advance. Since the LaTeX formula in the exercises follows a set of unified coding rules, we first replace LaTeX formulas uniformly through ontology replacement method, then perform unified preprocessing together with other text.

In the unified preprocessing of the formula text, the entities and attributes in the exercises are identified and replaced from entities to attributes. During the replacement process, the replaced entities or attributes and the replaced forms are saved in a dictionary. , That means, replace these formula texts $f = \{f_1, f_2, \ldots\ldots, f_M\}$ to obtain $\tilde{f} = \{\widetilde{f_1}, \widetilde{f_2}, \ldots\ldots, \widetilde{f_M}\}$. The partial replacement relationship is shown in Table 1.

Table 1. Formula replacement table.

f	$\tilde{f}$
complement	补集
sqrt	根式
^	幂
+	加号
cm	厘米
pi	圆周率

After the replacement, splice $\tilde{f}$ with the text $w = \{w_1, w_2, \ldots\ldots, w_M\}$ according to the original position, and get the text representation of the exercise $\widetilde{w}$.

$$\widetilde{w} = w \oplus \tilde{f} \quad (1)$$

For exercise representation $\widetilde{w}$, we apply python's Chinese word segmentation package Jieba for word segmentation. Jieba has three segmentation modes: precise mode, full mode and search engine mode. Here we utilize precise mode to accurately segment a complete sentence into independent words according to the segmentation algorithm. Then use a self-built stop word list to delete some words that cannot express specific meanings. This specific process is shown in Figure 2.

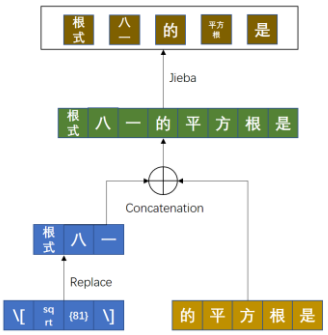


Figure 2. Example diagram of exercise formula preprocessing.

3.2.2 *Vector representation of multi-dimensional exercise*

In view of the superiority of Bert in the field of natural language processing, we exploit Bert to conduct self-supervised learning and training on the text features of exercises. We employ three embedding layers and a 12-layer encoder to pre-train the exercise representation. As shown in the question representation part in Figure 1, it includes three layers. The function of the token embedding layer is to convert each word segmentation into a 768-dimensional vector, then the segment embedding layer distinguishes differences between the vectors of two sentences, the position embedding layer can help understand the order of words. When all the embedding processes are done, we add the results of each layer element by element to get the input of the Bert encoder layer. After encoding by the 12-layer encoder, we can get the final text vector representation b_T.

$$b_T = e_{Token} + e_{Segment} + e_{Position} \quad (2)$$

In this way, semantic information of exercises can be obtained automatically without any extra expert manual coding.

3.2.3 *Representation of concepts*

In the dataset, each exercise will be marked with concepts involved, $k = \{k_1, k_2, \dots\dots, k_M\}$. With reference to EKT, we select the first concept k_1,which is also the most relevant one to represent the concept involved in exercise, then all the concepts of exercises are encoded into a vector of length |E| through one-hot vector encoding, E is the set of exercises, |E| represents the number of exercises, the encoded vector is adjusted by a layer of sigmoid activation function into the concepts representation c_T.

After getting the text representation and concepts representation, we concatenate them together as the final exercise embedded matrix.

$$x_T = b_T \oplus c_T \quad (3)$$

$\oplus$ means concatenate two vectors in a certain dimension, and the length of vector obtained after concatenating is |E|+768.

3.3 Exercises association modeling

After obtaining the final merged exercise embedding matrix, we need to model the entire exercising process of each student and obtain his hidden state at each step, it will be affected by both the history exercises sequence and his responses.

3.3.1 *Student response embedding*

First of all, we combine the student's response with each exercise representation. Specifically, at each step t,we combine exercise embedding x_T with the corresponding score r_T as the input to the recurrent neural network.

We first extend the score r_T to a feature vector $0 = (0, 0, \dots, 0)$ with the same dimensions of exercise embedding x_T and then learn the combined input vector $\widetilde{x_T}$ as:

$$\widetilde{x_T} = \begin{cases} x_T \oplus 0 & r_T = 1 \\ 0 \oplus x_T & r_T = 0 \end{cases} \quad (4)$$

After the concatenating, the student sequence becomes $s = \{\widetilde{x_1}, \widetilde{x_2}, \dots\dots, \widetilde{x_T}\}$.

3.3.2 *One-way time series knowledge tracing*

Like the original DKT method, a hidden layer is applied to track changes in student's knowledge status, the formula is as follows:

$$i_T = \sigma(W_i \cdot [h_{T-1}, \widetilde{x_T}] + b_i) \quad (5),$$

$$f_T = \sigma(W_f \cdot [h_{T-1}, \widetilde{x_T}] + b_f) \quad (6)$$

$$o_T = \sigma(W_o \cdot [h_{T-1}, \widetilde{x_T}] + b_o) \quad (7)$$

$$\widetilde{c_T} = tanh(W_c \cdot [h_{T-1}, \widetilde{x_T}] + b_c) \quad (8)$$

$$c_T = f_T \cdot c_{T-1} + i_T \cdot \widetilde{c_T} \quad (9)$$

$$h_T = o_T \cdot tanh(c_T) \quad (10)$$

Where c_T is the long-term state at time T, i_T, f_T, and o_T are the input gate, forget gate, and output gate in LSTM respectively. tanh represents the tanh activation function, $tanh(z_i) = (e^{z_i} - e^{-z_i})/(e^{z_i} + e^{-z_i})$, σ represents sigmoid activation function, $\sigma(z_i) = 1/(1 + e^{-z_i})$.

3.3.3 *Bidirectional time series knowledge tracing*

In order to better obtain the association between the exercises, we introduce a bidirectional long and short-term memory neural network [14] to obtain the hidden state representation of the students, because Bi-LSTM can make full use of the exercises representation in both forward and backward directions [15], it can obtain the association between the exercises. Specifically, after getting $s = \{\widetilde{x_1}, \widetilde{x_2}, \dots\dots, \widetilde{x_T}\}$, we set the input of the first layer of LSTM to $\overrightarrow{\boldsymbol{h}}^{(0)} = \overleftarrow{\boldsymbol{h}}^{(0)} = \{\widetilde{x_1}, \widetilde{x_2}, \dots\dots, \widetilde{x_T}\}$, at each time T, the forward hidden state of each layer $(\overrightarrow{h_T^{(l)}}, \overrightarrow{c_T^{(l)}})$ and backward hidden state $(\overleftarrow{h_T^{(l)}}, \overleftarrow{c_T^{(l)}})$ updates with the input from previous layer from each direction. The specific formula is as follows:

$$\overrightarrow{h_T^{(l)}}, \overrightarrow{c_T^{(l)}} = \text{LSTM}\left(\overrightarrow{h_T^{(l-1)}}, \overrightarrow{h_{T-1}^{(l)}}, \overrightarrow{c_{T-1}^{(l)}}; \overrightarrow{\theta}_{\text{LSTM}}\right) \quad (11)$$

$$\overleftarrow{h_T^{(l)}}, \overleftarrow{c_T^{(l)}} = \text{LSTM}\left(\overleftarrow{h_T^{(l-1)}}, \overleftarrow{h_{T+1}^{(l)}}, \overleftarrow{c_{T+1}^{(l)}}; \overleftarrow{\theta}_{LSTM}\right) \quad (12)$$

The association between exercises can be captured by Bi-LSTM. Since the hidden state of each direction only contains the association of one direction, it is beneficial to combine the hidden state of both directions together to obtain the final student hidden state representation:

$$H_T = \text{concatenate}\left(\overrightarrow{h_t^{(L)}}, \overleftarrow{h_t^{(L)}}\right) \quad (13)$$

3.4 Student performance prediction

After the above steps, we get the student's hidden learning state sequence $\{H_1, H_2, \dots\dots H_T\}$ and the exercise sequence $\{x_1, x_2, \dots\dots x_T\}$, both of which will affect the student's final answer. We utilize two layers of bidirectional neural networks to obtain the predicted student performance, as shown in the formula:

$$y_{T+1} = Tanh(W_1 \cdot [H_T \oplus \widetilde{x_{T+1}}] + b_1) \quad (14)$$

$$\widetilde{r_{T+1}} = \sigma(W_2 \cdot y_{T+1} + b_2) \quad (15)$$

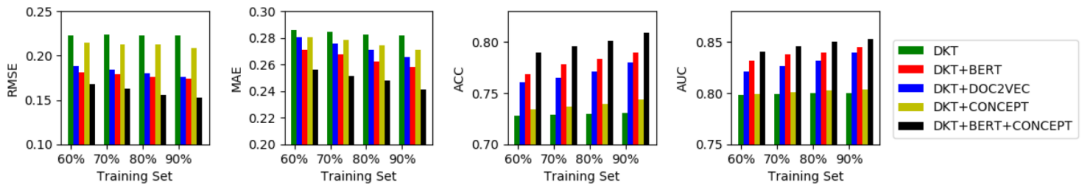


Figure 4. Four indicators performance after adding multi-dimensional feature.

The first layer uses the Tanh activation function, and the second layer uses the sigmoid activation function. After two layers, the final prediction result $\widetilde{r_T}$ is obtained. It is a scalar, which represents the probability of answering the question e_T correctly.

3.5 Training and optimization

The method is optimized by conducting the binary cross-entropy loss function, which calculates the loss between the true response r_T and the probability of correct answer $\widetilde{r_T}$, and adjusts the model parameters such as exercise embedding parameters and student response embedding by inverse transfer until the value converges. The loss function is defined as:

$$\mathcal{L} = -\sum (r_T \log \widetilde{r_T} + (1 - r_T) \log(1 - \widetilde{r_T})) \qquad (16)$$

4. EXPERIMENTS

In order to ensure the reliability of the experimental results, we carry out several baseline comparison experiments on a real dataset. This section will focus on the selection of data set and the comparison of benchmark models, as well as a discussion of the final experiments on a real dataset. This section will focus on the selection of data set and the comparison of benchmark models, as well as a discussion of the final experimental results.

4.1 Dataset

The method we proposed was validated on a dataset called Eanalyst-math. The data comes from a widely used evaluation system in China [16], from offline to online, that selects elementary school math exercises as experimental subjects. The data collected by EAnalyst-math mainly includes homework, unit tests, and term tests. Each assignment or evaluation is regarded as a collection of exercises, which is more in line with the actual education situation in China. EAnalyst-math recorded a total of 525,638 interactions from 1,763 students, with an average of 298.1 responses per student.

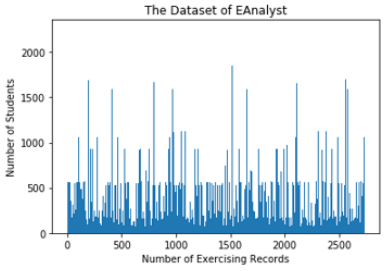


Figure 3. Student-interaction distribution diagram of the Eanalyst-math dataset.

4.2 Settings

In order to predict students' future response, we can evaluate it by classification and regression respectively [17]. From the perspective of classification, the area under the Receiver Operating Characteristic (ROC) curve AUC and the prediction accuracy ACC are used to measure the prediction performance. From a regression point of view, we choose Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to quantify the distance between the predicted result and the actual response.

Each data set is divided into 7:3 based on students, 70% is utilized for training verification, and 30% for testing. In order to avoid the contingency of the evaluation results, we implement the standard five-fold cross-validation division for all models and all training validation subsets, that is, 80% training set and 20% validation set, and exploit the average value as the final comparison result.

There are many hyperparameters in the model, among which the number of hidden units (h), batch_size (b) and learning rate (l) will have a greater impact on the results. We conducted many experiments to explore the influence of the changes of these hyperparameters on the performance of the model, and finally found that the performance results were optimal when l=0.09, h=16, and b=16.

4.3 Results

4.3.1 Accuracy comparison

We compare our ERAKT with three other baselines on the dataset. The experimental results are shown in Table 2. In general, ERAKT has significantly improved AUC and ACC results, MAE and RMSE results are significantly lower, which proves that the performance of ERAKT is better than the others. Especially, our ERAKT performs better the EERNN, a state-of-the-art model which including the exercise content information. Next, the exercise-aware methods (i.e. EERNN and ERAKT) outperform other models that ignore the exercise content (i.e. DKT and DKVMN). This experimental result validates the conclusion of EKT [11].

Table 2. Accuracy comparison

Method	AUC	ACC	MAE	RMSE
DKT	0.79	0.7301	0.2827	0.2233
DKVMN	0.8783	0.8072	0.2678	0.1346
EERNN	0.8836	0.8213	0.2495	0.131
ERAKT	0.9025	0.8407	0.2203	0.1278

4.3.2 The influence of multi-dimensional features of exercises on the prediction results

In view of the fact that multi-dimensional features will affect the performance of knowledge tracing, we explore the impact of different features on the performance of the model by adding them into the model. The results are shown in Table 3. It can be seen that the effect of ERAKT, which integrates multi- dimensional features, is significantly better than other models which only add semantic features or concepts features. It is worth mentioning that all the models we mentioned above perform better than the original DKT model.

Table 3. The influence of multi-dimensional features of exercises on the prediction results

None	DKT	0.79
Semantics	DKT+Doc2Vec	0.8266
	DKT+Bert	0.8325
Concept	DKT+Concept	0.83
Multi-dimensional features	DKT+Bert+Concept	0.8463

From a semantic point of view, no matter which method (Doc2Vec or Bert) is adopted to obtain the exercise representation, the results after embedding the method have been greatly improved, which shows that the text content of the exercises does have a non-negligible impact on the prediction result. From the perspective of concepts, the addition of conceptual features, while not much improved, was still about 1% higher than the original knowledge tracking method.

In order to better eliminate the influence of the data set division ratio on the results, 60%, 70%, 80% and 90% of the data set are applied as the training set, and the rest as the test set. As shown in Figure 4, the result is consistent with the performance of 70% division, with both AUC and ACC increasing and RMSE and MAE decreasing as the data set increases, which proves that the increase in the training set can Enhance forecasting effect.

4.3.3 The impact of association on the prediction results

After the experiment proved that integrating exercises representation can improve the accuracy of prediction, we then designed a comparative experiment to explore the impact of exercises association in predicting.

Table 4. The influence of exercise association on prediction results

RNN	LSTM	GRU	BI-LSTM
0.8463	0.8543	0.8637	0.9065

After we integrate exercises representation, we chose different time series modeling methods to model and predict student's responses. A comparison of the more commonly applied RNN [18], LSTM [19] and GRU [20] with the Bi-LSTM used in our method is shown in table 4. It can be seen that RNN is the worst performer, with LSTM and GRU in the middle, and Bi-LSTM the best. It proves that exercises association has a great influence on the prediction performance.

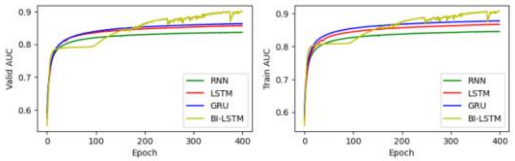


Figure 5. Auc and Loss convergence fluctuation diagram after adding exercise association

We have drawn the model convergence of the four time series modeling methods. It can be seen that the Bi-LSTM fluctuates slightly, and the model convergence curves of the other three methods are relatively smooth.

5. CONCLUSION AND DISCUSSION

In our proposed method, we propose a new context-aware KT method that integrates mathematical exercise representation and association of exercises, through which we can predict the performance of a student on the exercises, thus helping teachers to adjust their teaching plans dynamically.

Experiments have verified the effectiveness and reliability of our method. It can be seen from the experimental results that the prediction results are significantly improved after integrating multi-dimensional features and exercise association.

As for the exercise semantic representation, Bert can obtain more exercise information, which is better than Doc2Vec after integrating. This is because Bert realizes the processing of data of time series through the attention mechanism and it supports parallel computing, which is validated in Bert [8]. In the case of sufficient resources, the computing speed of Bert will be much faster than LSTM, and the residual network which inside of Bert can prevent the network structure from being too complicated. It makes the model perform better.

In the aspect of exercise association, in section 4.3.3, we use four different time series modeling methods, in which the Bi-LSTM exploited in ERAKT method has the best effect, then GRU's performance is relatively well among the remaining three. This is because of the internal structure of them. LSTM and GRU can solve the problem of long-term memory and can avoid the problem of gradient disappearance in RNN. Compared with LSTM, GRU can reduce the risk of over-fitting. Therefore, GRU has the best prediction performance and RNN is the worst, this conclusion can also be obtained in LSTM [19] and GRU [20]. But all of them three can not get the association between exercises.

The bidirectional structure of Bi-LSTM not only preserves the past information, but also the future one. Therefore, all the content can be effectively used to obtain the association between the exercises, which greatly improves the accuracy of prediction.

6. FUTURE WORK

At present, our research has achieved phased results, which can be applied in the actual teaching environment to assist teachers in teaching activities. Our future work will focus on two aspects:

(1) Explore knowledge tracking model that integrates multiple knowledge concepts, and at the same time integrate the sequence association between them.

(2) Show the students' mastery of each knowledge concept systematically. So as to to improve the accuracy of prediction, systematically promote it to facilitate the teaching work of teachers.

7. REFERENCES

[1] Yudelson M V, Koedinger K R, Gordon G J. Individualized bayesian knowledge tracing models[C]//International conference on artificial intelligence in education. Springer, Berlin, Heidelberg, 2013: 171-180.

[2] Piech C, Bassen J, Huang J, et al. Deep knowledge tracing[J]. Advances in neural information processing systems, 2015, 28: 505-513.

[3] Zhang, Jiani & Shi, Xingjian & King, Irwin & Yeung, Dit-Yan. (2017). Dynamic Key-Value Memory Networks for Knowledge Tracing. 765-774. 10.1145/3038912.3052580.

[4] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[5] Le Q, Mikolov T. Distributed representations of sentences and documents[C]//International conference on machine learning. 2014: 1188-1196.

[6] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.

[7] Joulin A, Grave E, Bojanowski P, et al. Bag of tricks for efficient text classification[J]. arXiv preprint arXiv:1607.01759, 2016.

[8] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[9] VASWANI, Ashish, et al. Attention is all you need. In: Advances in neural information processing systems. 2017. p. 5998-6008.

[10] Su Y, Liu Q, Liu Q, et al. Exercise-enhanced sequential modeling for student performance prediction[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2018, 32(1).

[11] Huang Z, Yin Y, Chen E, et al. Ekt: Exercise-aware knowledge tracing for student performance prediction[J]. IEEE Transactions on Knowledge and Data Engineering, 2019.

[12] Sonkar S, Waters A E, Lan A S, et al. qDKT: Question-centric Deep Knowledge Tracing[J]. arXiv preprint arXiv:2005.12442, 2020.

[13] Ghosh A, Heffernan N, Lan A S. Context-aware attentive knowledge tracing[C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020: 2330-2339.

[14] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015).

[15] Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bidirectional lstm-cnns-crf. arXiv preprint arXiv:1603.01354 (2016).

[16] Huang, T., Li, Z., Zhang, H., Yang, H., & Xie, H. (2020). EAnalyst: Toward Understanding Large-scale Educational Data. *EDM*.

[17] J. Fogarty , R. S. Baker, and S. E. Hudson. Case studies in the use of roc curve analysis for sensor-based estimates in human computer interaction. In Proceedings of Graphics Interface 2005, pages 129–136. Canadian Human-Computer Communications Society , 2005.

[18] Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." arXiv preprint arXiv:1409.2329 (2014).

[19] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[20] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).

[21] S. Wang, J. Tang, Y. Wang and H. Liu, "Exploring Hierarchical Structures for Recommender Systems," in IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 6, pp. 1022-1035, 1 June 2018, doi: 10.1109/TKDE.2018.2789443.

[22] Tong H , Zhou Y , Wang Z . HGKT : Introducing Problem Schema with Hierarchical Exercise Graph for Knowledge Tracing[J]. 2020

[23] Chen P, Lu Y, Zheng V W, et al. Prerequisite-driven deep knowledge tracing[C]//2018 IEEE International Conference on Data Mining (ICDM). IEEE, 2018: 39-48.

[24] Shalini Pandey and George Karypis. 2019. A self-attentive model for knowledge tracing. In Proc. International Conference on Educational Data Mining. 384–389.

[25] Nakagawa H , Iwasawa Y , Matsuo Y . Graph-based Knowledge Tracing: Modeling Student Proficiency Using Graph Neural Network[C]// IEEE/WIC/ACM International Conference on Web Intelligence. ACM, 2019.

Gaining Insights on Student Course Selection in Higher Education with Community Detection

Erla Guðrún Sturludóttir*
Reykjavík University
Menntavegi 1
102 Reykajvík, Iceland
erlas13@ru.is

Eydís Arnardóttir
Reykjavík University
Menntavegi 1
102 Reykajvík, Iceland
eydis13@ru.is

Gísli Hjálmtýsson
Reykjavík University
Menntavegi 1
102 Reykajvík, Iceland
gisli@ru.is

María Óskarsdóttir†
Reykjavík University
Menntavegi 1
102 Reykajvík, Iceland
mariaoskars@ru.is

ABSTRACT

Gaining insight into course choices holds significant value for universities, especially those who aim for flexibility in their programs and wish to adapt quickly to changing demands of the job market. However, little emphasis has been put on utilizing the large amount of educational data to understand these course choices. Here, we use network analysis of the course selection of all students who enrolled in an undergraduate program in engineering, business or computer science at a Nordic university over a five year period. With these methods, we have explored student choices to identify their distinct fields of interest. This was done by applying community detection (CD) to a network of courses, where two courses were connected if a student had taken both. We compared our CD results to actual major specializations within the computer science department and found strong similarities. Analysis with our proposed methodology can be used to offer more tailored education, which in turn allows students to follow their interests and adapt to the ever-changing career market.

Keywords

Community detection, higher education, Louvain method, bipartite networks, student network, course selection

1. INTRODUCTION

University students enter higher education with a plethora of courses to choose from on their path to graduation. Gaining insight into student choices holds significant value for universities, especially those who aim for flexibility in their programs and those who wish to adapt quickly to changing demands of the job market. For example, the fast rise in popularity of machine learning over the past years could impel universities to make machine learning and related courses readily available to their students. In contrast, more subtler trends could be directly identified by the students' choices rather than an obvious shift in the job market.

Numerous studies based on questionnaires and surveys have found that there are various components that contribute to a student's course selection [2, 19, 20]. These are factors such as learning value, workload, age and academic performance [2]. Of these, the learning value of the course (which refers to factors such as intellectual level and interest in the topic) has been found to be the most influential factor in course selection. Course selection has also been a target in studies aiming to understand the gap between student mindsets and career demands [20]. Maringe [19] found that although intrinsic interest was important, course choices depend mainly on future career goals. According to the author, universities may need to adapt their strategies to the idea that students' course choices now seem to reflect their expectations of future employment rather than simply interests. Thus, universities would benefit greatly from a deeper understanding of the path their students choose towards their degree.

Educational data mining (EDM) has risen as a new field to answer these and other questions about students and their learning environment. It utilizes a variety of analytical methods and applies them to the vast amounts of data that has become available with increased digitization of administrative educational information. For example, EDM methods have already been applied to try to accurately predict college success using common classification algorithms with different feature sets [31]. They have also been used to analyze student clicking behavior in online courses to determine students' learning strategies and how those strategies can have an impact on their learning outcomes [1], as well as to predict student dropout [10]. One area of educational stud-

*EGS and EA contributed equally.

†Corresponding author.

Erla Guðrún Sturludóttir, Eydís Arnardóttir, Gísli Hjálmtýsson and María Óskarsdóttir "Gaining Insights on Student Course Selection in Higher Education with Community Detection". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 367-374. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

ies that has not received much attention is student course selection, despite its importance in understanding student interests and preparing them for a future career [28].

In this paper, we aim to reveal patterns in course selection through EDM, providing a new data-driven technique based on institutional analytics to gain insight into students' interests that would otherwise be difficult to discern. This knowledge can then be used for monitoring student interests and ensuring that courses reflecting those interests are available. We examine whether network analysis applied to students' course data, with a focus on community detection (CD), can effectively be used to identify university students' fields of interest. To accomplish this, we use a weighted projection network in combination with CD to explore student course selection. We focus on communities of elective courses for different majors and compare them to some of the official specializations the university already has to offer. Deeper understanding of students' choices is a stepping stone into allowing students to take more control over their studies, improve flexibility in the curricula, and facilitate students' pursuit of their interests.

2. RELATED WORK

A promising method for EDM is to represent educational data as networks. In general, networks consist of nodes and edges, where the nodes can for example represent people, countries or cells, and edges represent connections between nodes based on factors such as spatial and temporal proximity or social connections such as friendships [12, 8]. Network analysis is used to look at internal characteristics and the connections and patterns of nodes and edges, providing the ability to better understand the fundamental structure of networks and the real-life phenomena they model [29]. Different methods can be used to analyze networks, for example by looking at structural characteristics such as centrality, which indicates the importance of any given node in the network by assuming nodes that are more central have higher control over information passed through the network [8]. Community detection is another common way of analyzing networks which allows for the aggregation of different nodes into communities based on shared characteristics by identifying groups of nodes that have a high number of edges within themselves but fewer edges to other groups [12].

A common application of network analysis in educational settings is to understand social connections between students. This has helped reveal the negative effects of student interdependence in music education programs and its relationship to the program's friendship networks [26], as well as identifying how positive and negative friendship ties emerge [27]. Network analysis has also helped clarify the relationship between students' social networks and the development of their academic success [6, 14]. Furthermore, looking at students' social networks over time, close coequal communities are typically formed early on [30], although in some cases, students enhance their performance due to social relations outside their assigned group [24].

Although students' social networks have been studied, the exploration of students' course choices through network analysis has few precedents. Within the EDM field, Kardan et al. [16] used neural networks to predict course enrollment based on various factors such as course and instructor characteristics, and course difficulty. Further, Turnbull and O'Neale [28] used network analysis with CD and entropy measures to explore enrollment in STEM courses at the high school level. Among other results, they revealed that indigenous populations showed higher levels of entropy in their enrollment patterns, which was moderated by adolescent socioeconomic status. Neither of these studies focused on detecting student interests from course selection patterns.

3. METHODS

3.1 Data Source

Here, we use student and course data from Reykjavík University (RU). The university offers many different areas of study, including preliminary studies, undergraduate and graduate degrees. Most RU students are undergraduate students, and the RU undergraduate programs also offer the most variety of courses. Generally, the majority of RU undergraduate programs' courses are mandatory. These are the core courses each department decides is essential to their study program. The rest of the courses are either free choice electives, which can be any course in the university that the student qualifies for, or restricted elective courses from a selection tailored to the specific major.

We sample data from all graduated RU students that enrolled in the year 2014 or later and completed undergraduate programs in engineering, business, or computer science (CS) before 2021 (the total number of students was 1481). The university offers other programs as well, but we left them out since they have fewer students. The variables we look at include the student's registration ID and registration semester, the name and semester of each course a student has completed, and whether they passed or failed the course. We also include each student's department, major, and type of study (undergraduate, graduate, etc.).

To anonymize the data, we remove anything that could identify students, specifically their social security number and a numerical registration ID and give them a unique random sequence of numbers to replace both original numbers. For each student, we also remove any courses that they had de-registered from early in the semester. Further, for each major, courses taken by fewer than 5% of students are considered outliers and removed.

3.2 Network Analysis

3.2.1 Bipartite networks

We apply network analysis to the data to explore the fields of interests of RU students from a data driven perspective. Many real-world networks have a bipartite structure, where nodes belong to one of two groups or divisions and edges connect nodes of opposite groups without within-group edges [3]. In our bipartite network, the students make up one division of the nodes, and courses the other. If a student has taken a course, an edge is created between the respective nodes. Since edges represent that a student has taken a course, there is no edge between two students nor between two courses (see Figure 1, left).

Although bipartite networks give a more realistic and detailed representation of the system, analyzing them can be

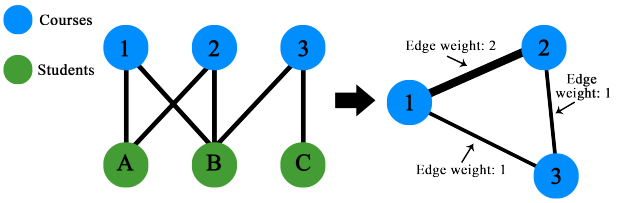


Figure 1: From bipartite network to weighted projected network. Left: a bipartite network, where the blue nodes represent courses and the green nodes, students. Right: a unipartite network has been obtained from the bipartite network, where the nodes are courses and the edges have weights that determine how many students have taken both courses.

complex. Therefore we project the bipartite network onto its unipartite counterpart (see Figure 1, right) [3]. This leaves a network with one type of nodes that can be analyzed with typical network methods. The resulting projected network consists of nodes representing the courses and edges between two nodes indicating that a student has taken both courses. We assign weights to the edges to represent the number of students who have taken both courses (see Figure 1).

A base problem with projection of bipartite networks is that a lot of important information in the original bipartite network is lost. Thus, we may end up connecting all courses in the network to each other –and form a clique– as long as they have at some point been taken by the same student, without taking into account how many students connected the two courses in the original bipartite network. Here, we address this by assigning weights to the edges in the projected network [3], where the weights represent the number of students who have taken both courses (see Figure 1).

3.2.2 *Community detection*

Building on the weighted projected networks, we use CD with the objective of inferring fields of interests in students' course selection. To identify fields of interest, we want to emphasise electives. However, in our data set, the information on which courses are mandatory and which are electives is incomplete. Mandatory courses along with very popular electives appear in the network as hubs, which usually occur in real-world networks as nodes with much higher degrees and edge weights than the other nodes [4]. We therefore define hubs in a data driven way, where a node is a hub if its total edge weight is at least one standard deviation above the mean edge weight of all nodes. We remove hubs from the network based on this definition.

Next, we apply the Louvain algorithm for CD [7]. This is an established, computationally efficient, fast converging method that produces accurate communities with high network modularity, especially in smaller networks [7, 17, 12, 23]. It has been successfully applied to identify communities of intrinsic brain systems [9], and to help create friend lists for Facebook users [18]. Modularity, is a measure of edge density within a partition (or proposed community) as opposed to edge density between partitions, whereby a higher modularity suggests a more cohesive community, separate from the others in the network. Importantly for our analysis using weighted projected networks, the Louvain algorithm can be used both with weighted and unweighted edges. The method starts by assigning each node to its own community [7], as seen in Figure 2. It then iterates over all nodes of the network and assesses the modularity gain obtained by assigning the node to the same community as each of its neighboring nodes. Next, the node is assigned to the community that yields the largest positive modularity gain, or maintains its current community if no positive modularity gain can be achieved by switching communities. This way, each new community assignment brings us closer to optimal modularity. The nodes are usually considered multiple times and the final iteration is determined when no switch leads to a gain of modularity, resulting in optimal partitioning of the network. This optimal partitioning is a local maxima, as the result is influenced by which node is considered first and the order in which nodes are visited. For some communities, we re-apply the Louvain algorithm for more detailed results, while using the inter/intra weight density ratio described below to ensure our communities maintain high quality.

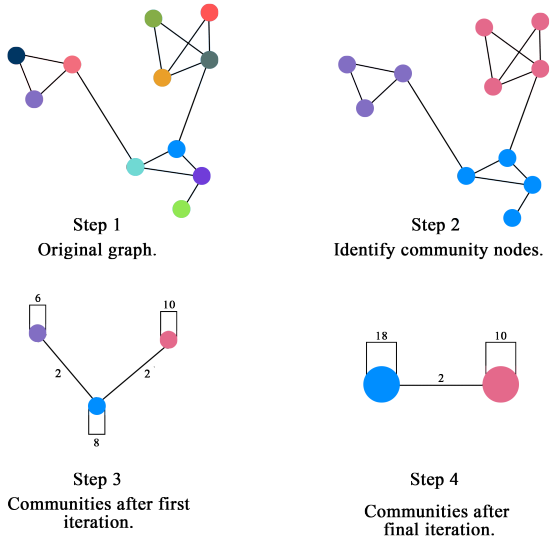


Figure 2: The Louvain algorithm. The first step of the algorithm is to assign each node to its own community. In step 2, a random node is selected to start the community aggregation process. All nodes are visited and allocated to the community of one of their neighbors or maintain their current community, depending on which choice gives the highest gain in modularity for the network. When no more modularity gain is possible in the network, step 3 is to aggregate the nodes of each community into new super-nodes. Here, the numbers given show the sum of node edges within and between supernodes. Steps 2 and 3 are then repeated until modularity has been optimized, as seen in step 4.

3.2.3 *Community validation*

Although the objective of CD is to split nodes into groups based on their connections within versus outside the group, there are many more aspects to consider [12]. One important factor is intra-cluster density, which refers to how many edges there are within the community as a ratio of how many possible edges there could be if all nodes of the community were connected to each other. This is contrasted by inter-

cluster density, which shows how many edges go from the community to the rest of the network as a proportion of the maximum possible connections. High intra-cluster density may suggest a strong and cohesive community, however if it coincides with equally high inter-cluster density, it may simply suggest a strong and cohesive overall network.

To assess the quality of our communities, we use intra and inter weight density [13]. This is the same as intra and inter edge density previously described, but now accounting for weighted edges. The two are defined as follows:

$$WD_{\text{inter}} = \frac{w_C^{\text{ext}}}{\bar{w} n_C (n - n_C)} \text{ and } WD_{\text{intra}} = \frac{w_C^{\text{int}}}{\bar{w} n_C (n_C - 1)/2},$$

where w_C^{ext} is the sum of edge weights connecting the community to the rest of the network, or external community edges. We divide this by the estimated total edge weight of the network, which shows the edge weight going from the community to the rest of the network as a proportion of the maximum possible edge weight (assuming that the average edge weight of the fully connected network were unchanged). Here, $\bar{w}$ is the average edge weight of the network, n is the total number of nodes in the network and n_C is the total number of nodes within the community. Similarly, w_C^{int} refers to the sum of edge weights inside the community, which is divided by the expected total edge weight within the community. We then use a ratio of these two measures (WD_{inter} / WD_{intra}) to obtain the community strength on a scale where 0 is the strongest value, indicating a community that is disconnected from the rest of the network, and a value of 1 indicates a community equally connected within itself as to the rest of the network. We call this measure *density ratio* and use it not only to determine the community strength, but also to ensure that as we create smaller and more focused communities, community strength is not compromised.

3.2.4 Comparing communities and specializations

To further assess the real-world application of the communities we detect, we compare them to specializations within RU's Computer Science (CS) department, described in Table 4 in the Appendix. Any student who pursues an undergraduate degree in CS at RU has the option to graduate with a specialization in a certain field. The specializations do not need to be declared at enrollment but any student who fulfills the requirements can choose to add this to their graduation certificate. The specializations offered are Artificial Intelligence, Law, Web- and User Experience (UX) Design, Sports Science, Game Development and FinTech. Each specialization has 2-4 core courses that students need to complete, along with 1-3 courses from a pool of specialization-specific electives. Our approach to defining fields of interest is purely through data driven CD. Comparing the detected communities with these specializations helps validate the results and perhaps provide a reference for the creation of new specializations. We compare both the courses in each community and specialization, and the number of students belonging to a specialization versus those belonging to the corresponding community. We define a student as belonging to a community if they have taken at least 50% of the community's courses, with a special case of two course communities where both courses have to be completed.

3.3 Tools

Aside from the initial retrieval and anonymization of data, which we do using C# and SQL, all code for the data analysis was written in Python 3.9. We use multiple Python libraries to help with the data analysis. For our network analysis, we mainly utilize the NetworkX library [21]. For more general data manipulation, we use the pandas library [22]. We used Gephi for the majority of our network visualization [5], along with the Matplotlib library [15].

4. RESULTS

4.1 Communities that Reflect Interest Fields

We conducted CD with the Louvain algorithm on three undergraduate majors: engineering, business, and computer science. These majors have quite different program structures and emphases on electives, with the business major having the lowest number of elective courses allowed in their study plan (four electives). This is followed by the CS major with 11 electives and finally engineering, which offers only four free electives but nine "guided electives" (that is, nine electives must be specific to engineering), depending on the chosen engineering specialization.

We first look at the communities for the engineering department, see Figure 4 and Table 2 in the Appendix, which after hub removal consisted of 81 courses taken by 496 undergraduate students. Reykjavík University offers various undergraduate engineering programs such as biomedical engineering, financial engineering, and mechatronics engineering. These engineering majors all fulfill the same core courses in addition to some additional major-specific requirements. These majors are quite structured and offer few free elective courses. Due to the similarity in the core courses of these programs, we group them together into a more general engineering major. This means that the hub removal method removed general core engineering courses but leave most specialty-specific courses in the network. The resulting engineering network has 81 course nodes and 2614 edges. The weighted average inter/intra weight density ratio is 0.24. This suggests that hub removal was effective and the average community is relatively strong. The communities we have detected were eight in total as seen in Table 2. Note that communities are named after common characteristics between the majority of the courses, even though rarely all courses of a community fall within that definition. As expected, these communities mainly correspond to the official engineering majors such as financial, biomedical, and electrical engineering, with electrical engineering being our strongest community (WD_{inter} / $WD_{\text{intra}} = 0.05$). However, we also observe unrelated communities that supersede the official majors, such as a community of applied design and another for business related courses not mandatory in the financial engineering major. Courses in these communities are commonly taken together by engineering undergraduates, suggesting a common interest not credited to the specialized majors.

There are 334 undergraduate students in our data set who majored in business. For this major, the network consists of 36 course nodes and 504 edges, with a weighted average WD_{inter} / WD_{intra} of 0.25, again suggesting strong communities, see Figure 5 in the Appendix. This is not unexpected, as the business major only allows electives in the final year,

giving business students less room to pursue distinct interests outside their core subjects. Table 3 in the Appendix shows the five communities identified within the business major. The strongest community is that of popular courses,

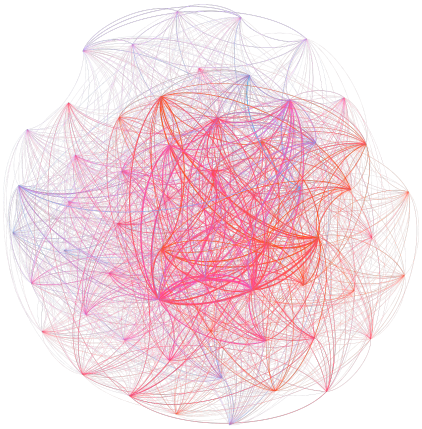


Figure 3: The network with communities for the BSc program in CS.

Table 1: Community detection results for BSc in CS.

	Community	No. courses	Density ratio
●	UX and Business	15	0.25
●	Engineering	13	0.17
●	Web and Software	10	0.20
●	Artificial Intelligence	7	0.39
●	Deprecated Courses I	6	0.08
●	Game Development	4	0.10
●	Deprecated Courses II	4	0.23
	Weighted average		**0.21**

which includes the most common electives in the business majors along with a handful of newer core courses (WD_{inter} / $WD_{\text{intra}} = 0.07$). These core courses were recently added to the study plan, meaning that they were only mandatory for a minority of the students in our data set. This is why these core courses were not identified as hubs and removed during hub removal. The business major also contains the weakest community of all the majors, management (WD_{inter} / $WD_{\text{intra}} = 0.71$). As the name suggests, this community includes various courses on management, such as service management and project management. The low inter and intra weight density ratio is interesting, as intuitively these courses would seem very connected. This is why measuring community strength is vital in determining the importance of the detected communities. The other business communities are both strong and reflect more specific interests, suggesting that there are students of the business major who actively seek distinct interests despite the program having no official specializations. The last major we explore is CS, with 377 students. Computer science has the least structured study plan of the three majors, as it puts a higher emphasis on unstructured flexibility and free electives. The CS course network consists of 59 nodes and 1492 edges. The communities (see Figure 3 are the strongest we found, with a weighted average WD_{inter} / WD_{intra} of 0.21. Most, but not all, detected communities seem to reflect an interest in a CS sub-field. However, the strongest community we have discovered was Deprecated Courses I (see Table 1), which represents older courses that may have been core courses at some point but are no longer being offered (WD_{inter} / $WD_{\text{intra}} = 0.08$). We conjecture that this community exists as some older students re-register to complete their undergraduate degree, for example after previously completing a CS diploma or taking a longer study break. It is therefore very intuitive that this specific sub-field is combined into our strongest community. Aside from communities based on deprecated courses, the other communities suggest that there is in fact an underlying pattern of interest fields present in the CS major, as observed for the other majors explored here.

4.2 RU Communities and Specializations

As a final validation of the communities we have detected for the CS undergraduate major, we now cross-reference our results with the actual specializations available for CS students. Unlike the other majors, CS offers a number of specializations meant to aid students in pursuing a specific subfield (see Table 4 in the Appendix for a short description of each specialization). However, only a subsection of students choose to do this. Of the students who graduated between 2014 and 2020, inclusive, only 9.5% fulfilled the requirements for a specialization. A further 13% partially fulfilled a specialization's requirements, by completing at least 60% of the specialization's core courses and 60% of the restricted electives needed.

Comparing the specializations and the communities we detected (shown in Table 1), we find interesting similarities. Our CD reveals that some communities are consistent with the specializations, but there is no absolute match. For the AI specialization (taken by 11 students, or 29% of those who graduated with a specialization), there is a partially corresponding community that includes both of the AI core courses (Artificial Intelligence and Machine Learning). There are 28 students who belong to this community, making it more popular than the official AI specialization. Although this community does not include any of the other courses from the specialization, it does include more theoretical and academically demanding courses than most other communities, suggesting a reflection of interest in theoretical computer science in general rather than specifically AI.

To fulfill the official AI specialization requirement, students must complete two core courses and three or more courses from a list of specialization-specific electives. However, in our data set most of these other electives were removed during either data cleaning (where we removed courses taken by fewer than 5% of students) or during hub removal and are therefore not part of any community. Interestingly, two of the remaining electives overlap between the AI specialization and that of Game Development. Both these courses have been sorted by our algorithm into a community that reflects Game Development much more strongly than AI, with 67 students. This is intriguing, as we know that students are much more likely to specialize in Artificial Intelligence than Game Development (only one student in our data set fulfills the requirements for Game Development), but this indicates that the gaming sub-field of Artificial Intelligence may be the

biggest area of interest for these students.

The final specialization for which we discovered a similar community is Web and UX design, which was by far the most popular specialization taken by students (with 23 students, or 64% of all students who had a specialization). While this specialization encompasses both web programming and user experience, the corresponding community of Web and Software Development (with 84 students) is much more web than UX specific. Most of the UX related courses belong to a separate community of 21 students that unites UX and business rather than UX and web design. This suggests that dividing the Web and UX design specialization into two distinct specializations (Web design and UX design) might be more appealing to students. Interestingly, the remaining four official specializations have no corresponding community in our results. This was to be expected, as these remaining specializations are very rarely pursued by students. That is, the communities we have detected are able to represent the specializations that students are actually choosing, but did not reflect other specializations. This is exactly what we expect of CD, with the added bonus of identifying fields of interests that may not have been previously considered.

5. DISCUSSION AND CONCLUSION

With this project, we aimed to find whether CD could be used to effectively identify students' fields of interest at RU. To maintain the scope of the results, we have presented only the findings for undergraduate majors in engineering, business, and CS. Our resulting communities vary slightly in strength and size, yet almost all of them contain courses of a general theme that seem to indicate that they do in fact reflect fields of interest. This builds on the results found by Turnbull and O'Neale [28], who performed CD on a similar school course network, but without hub removal. This resulted in much more general course communities that demonstrated important but slight differences in the overall majors. In focusing on fields of interests, removing the hubs has allowed us to increase the granularity of the resulting communities while still maintaining community strength and cohesion. However, one of the commonalities between these majors is that the largest community detected usually included the major's most popular courses, be that electives or new mandatory courses our hub removal does not consider. As Fortunato [13] suggested, using the inter/intra weight density, we were able to evaluate the quality of the communities that were detected with the Louvain algorithm.

The communities we have discovered encapsulate various distinct areas of interests for the different undergraduate majors RU has to offer. Additionally, for the CS department, we have verified that the detected communities also reflect the main areas students choose to specialize in, which further validates our findings. To our knowledge, applying CD in this way and for this purpose has not been done before. This provides an exciting new tool for universities to better understand their students' aspirations.

In improving knowledge of student course selection, we provide academic institutions with more tools to increase study flexibility for their students. This knowledge can then be used to decide which courses the university wants to offer. This knowledge is also useful for academic counselors when helping students to discover their own field of interest. Based on previous studies, we assume that interest is the main motivation behind course choices [2, 19]. However, these communities may be based on other factors. Examining the characteristics of courses that make up different communities might reveal other factors that contribute to course selection, such as course difficulty, grading, teacher characteristics, and more [25, 2, 19].

Although we were able to successfully apply network analysis to our student and course data, there were a few setbacks. One drawback in our analysis is the fact that although RU's administrative data has largely been digitized, this has not always been done in the most structured and data-mining friendly way. For example, all information on specializations was retrieved directly from RU's website and formatted manually, as this information is not stored in the university's data warehouse. Reliable information on the mandatory courses of each major was also not available, which was why we decided to use data driven hub removal. Improving data availability, centrality and consistency is currently a priority at RU, but should also be considered by other universities wanting to take full advantage of EDM methods.

Our findings show that network analysis with CD is a useful tool in understanding students' course selection. The course choice patterns found here can still be explored further. For example, the current results are based on data from students who enrolled in the same program at different times. Thus any small changes in the program structure between years can introduce noise in the data. Looking at individual registration years, perhaps including a larger university with more students, could give clearer results. Further, it would be interesting to repeat the same analysis over separate periods to discover changes in interest fields over time. Finally, it was out of the scope of the current paper to analyze trends based on more detailed characteristics such as gender, age or grades. Augmenting the communities with these factors could for instance provide a tool to identify differences in choices made by students who graduate successfully and those who struggle more with their studies, perhaps yielding an opportunity for early intervention.

Educational data mining is an exciting new field with the potential to greatly influence educational institutions and their students going forward [11]. This project aimed to reveal how network analysis could be used to enhance student course selection by improved understanding of students' academic interests. Our analysis has successfully led to meaningful results that could easily be replicated by most interested universities with digitized information. Coupling this increased understanding of student interests with added academic support gives universities the tools to raise flexibility within majors while maintaining educational quality. Hopefully, this and other research in the field can be used to offer more tailored and student-led education, which in turns allows students to follow their interests and easily adapt to the ever-changing demands of the job market.

6. ACKNOWLEDGMENTS

We appreciate the contributions made by Kolbrún Eir Óskarsdóttir who had a big impact on the conceptualization and development of this research.

7. REFERENCES

[1] N.-J. Akpinar, A. Ramdas, and U. Acar. Analyzing student strategies in blended courses using clickstream data. In *Proceedings of the 13th International Conference on Educational Data Mining*, 2020.

[2] E. Babad and A. Tayeb. Experimental analysis of students' course selection. *British Journal of Educational Psychology*, 73(3):373–393, 2003.

[3] S. Banerjee, M. Jenamani, and D. K. Pratihar. Properties of a projected network of a bipartite network. In *2017 International Conference on Communication and Signal Processing (ICCSP)*, pages 0143–0147. IEEE, 2017.

[4] A.-L. Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987):20120375, 2013.

[5] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009.

[6] D. Blansky, C. Kavanaugh, C. Boothroyd, B. Benson, J. Gallagher, J. Endress, and H. Sayama. Spread of academic success in a high school social network. *PLoS ONE*, 8(2):e55944, 2013.

[7] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2020.

[8] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca. Network analysis in the social sciences. *Science*, 323(5916):892–895, 2009.

[9] M. Cole, D. Bassett, J. Power, T. Braver, and S. Petersen. Intrinsic and task-evoked network architectures of the human brain. *Neuron*, 83(1):238–251, 2014.

[10] G. Deeva, J. De Smedt, P. De Koninck, and J. De Weerdt. Dropout prediction in moocs: a comparison between process and sequence mining. In *International Conference on Business Process Management*, pages 243–255. Springer, 2017.

[11] G. Deeva, S. Willermark, A. S. Islind, and M. Oskarsdottir. Introduction to the minitrack on learning analytics. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, page 1507, 2021.

[12] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[13] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[14] N. Gitinabard, F. Khoshnevisan, C. F. Lynch, and E. Y. Wang. Your actions or your associates? predicting certification and dropout in moocs with behavioral and social features. *International Educational Data Mining Society*, 2018.

[15] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[16] A. A. Kardan, H. Sadeghi, S. S. Ghidary, and M. R. F. Sani. Prediction of student course selection in online higher education institutes using neural network. *Computers & Education*, 65:1–11, 2013.

[17] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[18] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove. Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference - IMC '11*, page 61. ACM Press, 2011.

[19] F. Maringe. University and course choice: Implications for positioning, recruitment and marketing. *International Journal of Educational Management*, 20(6):466–479, 2006.

[20] V. C. Milliron. Exploring millennial student values and societal trends: Accounting course selection preferences. *Issues in Accounting Education*, 23(3):405–419, 2008.

[21] NetworkX developer team. Networkx, 2014.

[22] T. pandas development team. pandas-dev/pandas: Pandas, Feb. 2020.

[23] X. Que, F. Checconi, F. Petrini, and J. A. Gunnels. Scalable community detection with the louvain algorithm. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 28–37. IEEE, 2015.

[24] B. Rienties and D. Tempelaar. Turning groups inside out: A social network perspective. *Journal of the Learning Sciences*, 27(4):550–579, 2018.

[25] R. Sabot and J. Wakeman-Linn. Grade inflation and course choice. *Journal of Economic Perspectives*, 5(1):159–170, 1991.

[26] M. Sarazin. Can student interdependence be experienced negatively in collective music education programmes? a contextual approach. *London Review of Education*, 2017.

[27] M. A. Sarazin. Disliking friends of friends in schools: How positive and negative ties can co-occur in large numbers. *Social Networks*, 64:134–147, 2021.

[28] S. M. Turnbull and D. O'Neale. Entropy of co-enrolment networks reveal disparities in high school stem participation. *ArXiv*, abs/2008.13575, 2020.

[29] B. Wellman. Network analysis: Some basic principles. *Sociological Theory*, 1:155, 1983.

[30] Y. Xu, C. F. Lynch, and T. Barnes. How many friends can you make in a week?: Evolving social relationships in moocs over time. *International Educational Data Mining Society*, 2018.

[31] R. Yu, Q. Li, C. Fischer, S. Doroudi, and D. Xu. Towards accurate and fair prediction of college success: Evaluating different sources of student data. In *13th International Conference on Educational Data Mining*, 2020.

APPENDIX

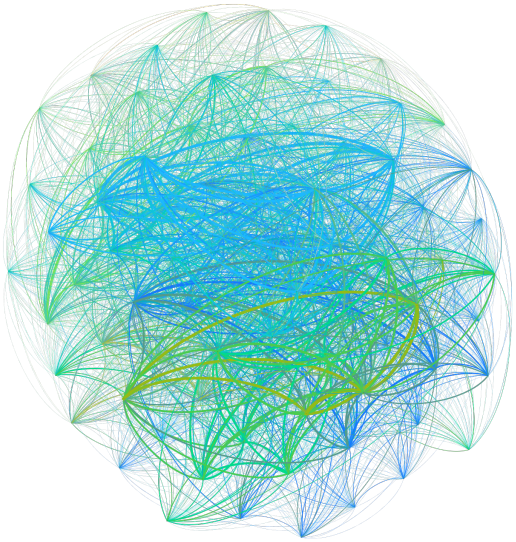


Figure 4: The network with communities for the BSc program in engineering.

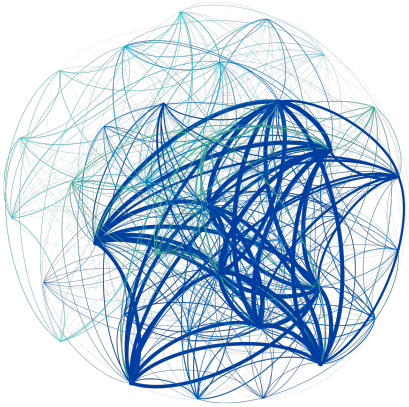


Figure 5: The network with communities for the BSc program in business.

Table 2: Community detection results for BSc in engineering.

	Community	No. courses	Density ratio
	Comp Sci and Mechatronics	25	0.37
	Engineering Management	15	0.16
	Finances and Management	10	0.25
	Biomedical Engineering	10	0.10
	Financial Engineering	9	0.21
	Electrical Engineering	5	0.05
	Applied Design	4	0.29
	Business	3	0.32
	Weighted average		**0.24**

Table 3: Community detection results for BSc in business.

	Community	No. courses	$WD_{\text{inter}}/WD_{\text{intra}}$
	Popular Courses	15	0.07
	Management	6	0.71
	Finance	6	0.29
	Operations	5	0.10
	Asset Management	4	0.36
	Weighted average		**0.25**

Table 4: Official specializations in the CS program.

Name	Description
Artificial intelligence	Core courses reflecting an interest in AI and machine learning, with electives focused on game development and analytical skills.
Game design	Core courses encompass game development in general, computer graphics and game engine architecture. Electives reflect more general programming skills and AI.
FinTech	Both core courses and electives focus on the financial part of the Financial Technology discipline, as all students taking these courses gain software development skills from the core courses of the CS major.
Web and UX design	As the name suggests, most courses for this specialization directly relate to either web programming (such as the courses Web Programming II and Web Services) or user experience (User-Focused Software Development, Human-Computer Interaction).
Psychology	Core courses in psychology that emphasize cognitive processing and research methodology. Any other psychology courses can then be chosen as electives.
Law	General law courses with some emphasis on intellectual property rights and negotiations.
Sports science	General sports science courses.

Automated Claim Identification Using NLP Features in Student Argumentative Essays

Qian Wan
Georgia State University
qwan1@gsu.edu

Scott Crossley
Georgia State University
scrossley@gsu.edu

Michelle Banawan
Arizona State University
mbanawan@asu.edu

Renu Balyan
SUNY at Old Westbury
balyanr@oldwestbury.edu

Yu Tian
Georgia State University
ytian9@gsu.edu

Danielle McNamara
Arizona State University
dsmcnama@asu.edu

Laura Allen
University of New Hampshire
Laura.Allen@unh.edu

ABSTRACT

The current study explores the ability to predict argumentative claims in structurally-annotated student essays to gain insights into the role of argumentation *structure* in the quality of persuasive writing. Our annotation scheme specified six types of argumentative components based on the well-established Toulmin's model of argumentation. We developed feature sets consisting of word count, frequency data of key n-grams, positionality data, and other lexical, syntactic, semantic features based on both sentential and suprasentential levels. The suprasentential Random Forest model based on frequency and positionality features yielded the best results, reporting an accuracy of 0.87 and kappa of 0.73. This model will be included in an online writing assessment tool to generate feedback for student writers.

Keywords

Argumentation, Claim identification, Argumentative writing

1. INTRODUCTION

Written argumentation has been an important area of study for many years [43, 45]. Recent developments in natural language processing (NLP) have introduced new approaches to automatically detect the discourse structure of argumentative essays [7, 8, 9, 10, 26, 33, 34, 38, 44, 45]. These studies have shown that content (i.e., lexical, syntactic, and semantic) and structural features (i.e., the positionality of tokens, sentences, and paragraphs) are effective in detecting discourse elements.

Researchers have used fixed discourse markers at the word and phrase levels [5, 12, 18, 42] as indicators of different argumentative structures. This approach has been applied in discourse [17, 19, 22] and NLP analyses [7, 8, 9, 47]. These studies generally identify relations between discourse markers and their functions according to the conceptual framework of conjunctive relations [36]. For instance, phrases such as *in summary* and *in conclusion* are associated with the discourse function of 'summarizing' an argument. Such discourse markers have been used to identify the attributes of the structural elements in argumentative essays [8, 9, 36, 37]. For example, Burstein et al. [7] annotated structural information of argumentative essays collected from TOEFL, GRE, and GMAT. Discourse markers indicating each of the argumentative functions were extracted automatically from the essays. A word list that contained the discourse markers and their corresponding argumentative functions was formed and used to automatically predict instances of argumentation. Similarly, Palau and Moens [37] implemented a context-free ruled-based approach for argumentation mining in legal texts. They focused on and developed rules based on common expressions encountered in the legal documents such as *for these reasons*, *in light of all the material*, and discourse markers, such as *however* or *furthermore*. Using this approach, they obtained accuracy of approximately 0.6 in detecting the argumentation structures, while maintaining F1-measure of around 0.7 for recognizing premises and conclusions in legal texts.

In more recent work, Stab and Gurevych [44, 45] provided publicly available corpora comprising students' argumentative essays and annotation guidelines for parsing argumentations. In these corpora, the essays were annotated based on three major argumentative categories: major claim, claim, and premise. They then used lexical, structural, syntactic, discourse markers, and other features to identify argument components. The lexical features consisted of binary lemmatized unigrams and the 2,000 most frequent bigrams extracted from a training corpus. The structural features captured the position of components in the text and the number of tokens in those components. Discourse markers included logical connectives such as *therefore*, *thus*, or *consequently* and the use of first-person pronouns (which indicated major claims). The syntactic features included part of speech (POS) distributions, number of sub-clauses, and the tense of the main verb. Using support vector machine models, Stab and Gurevych [45] found that a combination of all these features yielded an F1 score of 0.77. Khatib et al. in [3] employed a classifier for argumentativeness based on the research in [37, 44, 45], and evaluated its performance on student essays from [44]. Khatib et al. used n-grams, syntax, discourse makers and part of speech (POS) features in an argument. Their results indicated that a combination of n-grams, POS tags, and syntax features yielded accuracy of 0.64, 0.62, and 0.59 on classifying arguments in students' essays, while the full feature set model yielded an accuracy of 0.67. Though only unigram through tri-grams were included in the POS feature.

Though the use of discourse markers, n-grams and POS as indicators has been common in the detection of argumentative elements, few studies have examined whether using longer

Qian Wan, Scott Crossley, Michelle Banawan, Renu Balyan, Yu Tian, Danielle McNamara and Laura Allen "Automated Claim Identification Using NLP Features in Student Argumentative Essays". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 375-383. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

sequences of n-grams (beyond tri-grams) and their POS tags would contribute to identifying argumentative features. We also note that other types of linguistic features related to lexical, structural, cohesion, and affective features were not tested in previous studies [e.g., 37, 45]. Therefore, this study explores a wider range of NLP features, and examines their contribution to model accuracy. We do so specifically on a corpus of student essays annotated on theoretically-aligned classifications of argumentative elements expected in academic settings. This is in contrast to most of the existing corpora in English that are annotated for argumentative structures and are from the domains of law [e.g., 4, 37], biology and medicine [e.g., 20], and user-generated content, e.g., Wikipedia articles or debate data, see [1, 2, 27, 41]. Few corpora [44, 45] have been developed for argumentation mining in the educational settings. In this study, we build on Stab and Gurevych's work [44, 45] by developing a structurally annotated corpus based on the Toulmin model [46] of argumentation that better reflects the structure of student essays. Our objective is for the corpus – and the models of argumentation developed from the corpus – to contribute to the development of writing assessment tools that can deliver useful feedback to student writers.

Thus, in this study, we introduce a new corpus of essays annotated for argumentative features. We then develop NLP approaches to automatically identify claims in structurally annotated argumentative essays using length, frequency data of significant n-grams and POS tags, positionality data, and a wide range of lexical, syntactic, cohesion, and cognitive features extracted from a number of NLP tools [14, 15, 25, 24]. We compared the identification accuracy of multiple machine learning classifiers using different types of derived features at different levels (based on sentences or argumentative elements that are suprasentential). Our goal is to better understand whether and how the selection of the linguistic features, the level of units for identification (both sentential and suprasentential), and the choice of classifiers influence the accuracy of claim identification. Finally, we conduct an error analysis of the best model and discuss the distribution of the misclassification instances and related features. This study is guided by the following research question:

To what extent do length, frequency of significant n-grams (and POS tags of n-grams), lexical, syntactic, and semantic features, and positionality predict argumentative claims in essays?

2. METHOD

2.1 Corpus

For the analysis, we annotated 314 persuasive essays. The essays were written by undergraduate students (N = 314) at a public university in the United States who were native speakers of English. Two prompts from retired test banks of the Scholastic Assessment Test (SAT) were used. The prompts were counterbalanced such that half of the students wrote about 'originality and uniqueness' while the other half wrote about 'heroes versus celebrities.' All essays had been scored previously by expert raters for holistic writing quality. For each essay, we extracted the average number of letters per word, the number of words, number of types, type-token ratio, average number of words per sentence, the number of sentences and paragraphs. Descriptive statistics for these items of the 314 essays are reported in Table 1.

Table 1. Descriptive statistics of the persuasive essays

	Mean	SD	Median	Range
Letters per word	4.52	0.24	4.51	1.50
Number of words	354.46	118.20	344.00	680.00
Number of types	178.17	50.01	173.00	279.00
Type-token ratio	0.52	0.07	0.52	0.41
Words per sentence	17.74	4.30	17.06	35.08
Number of sentences	20.65	7.42	20.00	48.00
Number of paragraphs	3.86	1.38	4.00	7.00

2.2 Annotation of argumentative elements

The essays were structurally annotated by normed raters for argumentative elements. We used the modified Toulmin models [46] presented in [35] and [30] as the basis for the annotation rubric. The rubric adopted six elements (i.e., micro-categories) as the building blocks of the argumentation framework: Final Claim, Primary Claim, Counterclaim, Rebuttal, Data, and Concluding Summary. The definitions of each of these elements are presented in Appendix A.

The essays were coded by two annotators on the web-based text annotation platform 'Tagtog'[1]. The two annotators were both native speakers of English and were undergraduate students majoring in applied linguistics at a public university in the United States. Before independent annotation, a norming process was conducted to help ensure consistency in annotations. Once normed, the two annotators worked independently and coded the 314 essays in the opposite order to avoid recency effects.

The two annotators made decisions on both the boundary of an argumentative element and the category of the element. An argumentative element was inherently suprasentential (i.e., according to the annotation scheme derived from the norming session, it could contain one or more sentences, and the content could be over the span of paragraphs). Inter-rater reliability calculated using Fleiss's Kappa for all the annotations was 0.584 (p < 0.001), indicating fair to good agreement [16]. Disagreements of either boundary or category of the argumentative elements between the two annotators were adjudicated by an expert adjudicator who had years of experience teaching and conducting writing research. In the case of disagreement, the expert adjudicator compared the annotations from both annotators and made the final decision for both the boundary and the category of the argumentative element.

The current study focuses on the identification of claims versus non-claims, mainly because of the small sample size of the corpus and the distribution of micro-categories. Thus, we combined the categories of Final Claim, Primary Claim, Counterclaim, and Rebuttal into a single category of claims. The remaining categories of Data and Concluding summary were classified as non-claims as was any non-annotated text.

[1] https://www.tagtog.net

2.3 Training and test sets

Annotation of the data led to the classification of 2264 argumentative elements. As mentioned in Section 2.2, the argumentative elements were inherently suprasentential. We further split the elements into sentences to determine whether this influenced accuracy. All sentences from the same argumentative element were given the same annotation as the original category (i.e., claims or non-claims). We thus had two data sets: 1) a sentence-tokenized data set (N = 6326) and 2) a suprasentential data set (N = 2264). We randomly selected 70% of the argumentative elements as the training set, and the remaining 30% of the elements as the test set for both datasets. We report the number of argumentative elements, and number of claims and non-claims for the datasets in Table 2.

Table 2. Numbers of elements, claims and non-claims for the training and test sets

Data set	Number of elements	Number of claims	Number of non-claims
Suprasentential training set	1594	639	955
Suprasentential test set	670	267	403
Sentential training set	4401	935	3466
Sentential test set	1925	409	1516

2.4 Features

2.4.1 Word count

We extracted the number of words for each claim and non-claim at the sentential and suprasentential level.

2.4.2 N-gram frequency

We extracted n-grams and the POS combinations of these n-grams for both claims and non-claims. We assume that some n-grams (or POS n-grams) are more likely to identify claims versus non-claims (and vice versa), and the frequency of these key n-grams (or POS n-grams) could serve as good indicator of the type of an argumentative element or sentence. We used keyness values [21] as the measurement of importance of the n-grams or POS n-grams in claims and non-claims. Keyness values can provide evidence of whether n-grams and POS n-grams are more common in one corpus as compared with the other corpus. In the current study, we treated the claims and non-claims as two separate corpora.

Raw and normalized frequency (i.e., normalized by the total number of words in all claims and non-claims, respectively) for each n-gram (or POS n-gram) that occurred both in claims and non-claims were calculated. The keyness value of each n-gram was also calculated based on the frequency data following Rayson and Garside's guidelines [40]. Specifically, if an n-gram or POS n-gram had a keyness value greater than 3.84 (equivalent to $p < 0.05$), and if it had a higher normalized frequency in claims, it was considered more likely to occur in claims over non-claims, and vice versa. The range of the n-grams and POS n-grams was from unigram to seven-grams. NLTK [6] was used to tokenize the texts into n-grams and label the POS for the n-grams. For example, the following phrases *should be*, *would be*, *can be*, and *will be* were converted to the same POS n-gram combination: MD (modal) + VB (verb base). We did not remove stopwords before n-gram tokenization. For each suprasentential and sentential argumentative element in the training and test sets, we calculated the frequency of each type of the significant n-grams or POS n-grams (e.g., bigrams that were significant in claims), and normalized the frequency by the length (word counts).

2.4.3 Positionality of the elements

Beyond n-gram frequency, studies have shown that, the position of argumentative elements is an indicator of their structural function [e.g., 7, 8, 10]. In this study, two types of normalized positional variables for each argumentative element or sentence were calculated as positionality features.

Normalized element or sentence position in an essay was computed as the ratio of the element/sentence position in an essay to the number of elements/sentences in the essay (e.g., if an argumentative element or a sentence was the 5th element or sentence in an essay of 10 elements/sentences in total, the value of this variable would be 5 divided by 10, or 50%). The normalized position of the element or sentence in a paragraph was computed as the ratio of the element/sentence position in a paragraph to the total number of elements/sentences in that paragraph. That means, if an argumentative element or a sentence was the 2nd element (sentence) in a paragraph, in which there were 5 elements (sentences) in total, the value would be 2 divided by 5, or 40%).

2.4.4 Other lexical, syntactic, and semantic features

To explore whether additional lexical, syntactic, cohesion, and cognitive text features increased the accuracy in identifying claims and non-claims, we extracted 925 features for each of the argumentative elements. These features were extracted using the Suite of Automatic Linguistic Analysis Tools (SALAT) [14, 15, 25, 24]. SALAT includes multiple NLP tools including TAACO (Tool for the Automatic Analysis of Cohesion), TAALES (Tool for the Automatic Analysis of lexical Sophistication), TAASSC (Tool for the Automatic Analysis of Syntactic Sophistication and Complexity), and SEANCE (Sentiment Analysis and Cognition Engine). Two-sample t-tests or Wilcoxon's tests were conducted using the variables after removing SALAT variables that were not normally distributed. We then removed those variables where the results of t-test or Wilcoxon's test were not significant between the group of claims and non-claims. Finally, by visual inspection, 20 out of 131 variables that were relevant to argumentative elements were selected. Hand selection of variables was done to avoid problems of overfitting. The selected NLP features and their descriptions are presented in Appendix B.

2.4.5 Feature reduction

To avoid multicollinearity, we conducted correlation analyses among all the derived features (one versus all) for the two training sets, respectively. If two or more variables correlated with $r >$ 0.699, the variable(s) with the lower correlation with the category of the argumentative element/sentence were removed, and the variable with the higher correlation was retained. The feature reduction process was done on the two training sets first and then applied to the test sets. After feature reduction, the frequency features that were retained included word count (of the argumentative element or sentence), the frequency of the significant unigram in claims and in non-claims, bigrams and quad-grams in claims, and the frequency of significant POS unigrams, trigrams, four-grams, five-grams in claims and in non-claims, and frequency of significant six-grams in claims. The two positionality features and the selected 20 SALAT features were also retained.

Table 3. Model accuracy results

Classifier	Model	Accuracy	Kappa	Label	Precision	Recall	F1
Logistic Regression	Suprasentential - Frequency and positionality	0.852	0.691	Non-Claim	0.874	0.881	0.878
				Claim	0.818	0.809	0.814
	Suprasentential - Full features	0.845	0.675	Non-Claim	0.867	0.876	0.872
				Claim	0.810	0.798	0.804
	Sentential - Frequency and positionality	0.802	0.216	Non-Claim	0.817	0.965	0.885
				Claim	0.604	0.198	0.298
	Sentential - Full features	0.800	0.244	Non-Claim	0.823	0.951	0.882
				Claim	0.569	0.242	0.340
Naive Bayes	Suprasentential - Frequency and positionality	0.769	0.485	Non-Claim	0.747	0.931	0.829
				Claim	0.833	0.524	0.644
	Suprasentential - Full features	0.819	0.618	Non-Claim	0.831	0.878	0.854
				Claim	0.799	0.730	0.763
	Sentential - Frequency and positionality	0.791	0.267	Non-Claim	0.834	0.925	0.878
				Claim	0.515	0.301	0.380
	Sentential - Full features	0.789	0.271	Non-Claim	0.833	0.916	0.872
				Claim	0.506	0.318	0.390
K-Nearest Neighbors	Suprasentential - Frequency and positionality	0.836	0.650	Non-Claim	0.835	0.906	0.869
				Claim	0.837	0.730	0.780
	Suprasentential - Full features	0.787	0.526	Non-Claim	0.760	0.943	0.842
				Claim	0.865	0.551	0.673
	Sentential - Frequency and positionality	0.818	0.286	Non-Claim	0.827	0.973	0.894
				Claim	0.709	0.245	0.364
	Sentential - Full features	0.804	0.196	Non-Claim	0.813	0.976	0.887
				Claim	0.654	0.166	0.265
Support Vector Machines	Suprasentential - Frequency and positionality	0.863	0.714	Non-Claim	0.886	0.886	0.886
				Claim	0.828	0.828	0.828
	Suprasentential - Full features	0.833	0.652	Non-Claim	0.865	0.856	0.860
				Claim	0.786	0.798	0.792
	Sentential - Frequency and positionality	0.818	0.336	Non-Claim	0.839	0.951	0.891
				Claim	0.639	0.325	0.431
	Sentential - Full features	0.822	0.320	Non-Claim	0.833	0.968	0.896
				Claim	0.706	0.281	0.402
Random Forest	Suprasentential - Frequency and positionality	0.873	0.734	Non-Claim	0.886	0.906	0.896
				Claim	0.853	0.824	0.838
	Suprasentential - Full features	0.866	0.720	Non-Claim	0.890	0.886	0.888
				Claim	0.829	0.835	0.832
	Sentential - Frequency and positionality	0.832	0.419	Non-Claim	0.858	0.943	0.898
				Claim	0.664	0.421	0.515
	Sentential - Full features	0.829	0.390	Non-Claim	0.850	0.951	0.897
				Claim	0.672	0.377	0.483

To examine whether adding the SALAT features improved the accuracy of claim identification, we created two versions of the feature sets. The first version comprised the n-gram frequency (including word count) features and positionality features, and the second version comprised all the features (including the SALAT NLP features). Combined with the different levels of discourse units (sentential and suprasentential), four pairs of datasets (training and test sets) were prepared for modeling: the frequency and positionality versions along with the full feature versions at both the sentential and suprasentential levels.

2.4.6 Classifiers

We used the 'caret' [23], 'randomForest' [28], 'e1071' [32], and 'tidyverse' packages [48] in R [13] to apply Logistic Regression, Naïve Bayes, K-Nearest Neighbors, Support Vector Machines, and Random Forest models. 10-fold cross validation with five repeats was used. We trained and tested the four versions of data separately. For the SVM classifier, a linear, polynomial, and radial kernel was applied. The model with the best performance was selected to make predictions on the test set.

3. RESULTS

3.1 Model evaluation

The classification performances (precision, recall, F1 scores, accuracy, and Cohen's kappa) of the multiple models on the test sets are reported in Table 3.

Overall, the models developed on frequency and positionality features slightly outperformed the models developed using all the features. This indicates that adding lexical, syntactic, cohesion, and cognitive NLP features does not improve the accuracy of the classification of claims and non-claims. In terms of the selection of the unit of classification, the suprasentential models outperformed the sentential models. Finally, the suprasentential Random Forest

model based on frequency and positionality features yielded the best accuracy (0.873) and Kappa (0.734), followed by the suprasentential model based on the full feature set, which yielded an accuracy of 0.866 and Kappa of 0.720, which represents good performance based on the scale of Cohen's Kappa values [11].

3.2 Important variables

Variable importance for the best model (the suprasentential Random Forest model based on word count, n-gram frequency and positionality features) was reported by the 'caret' package. Table 4 shows the top 10 important variables and their importance values for this model.

The variable importance values showed that the length (word count) of an argumentative element, the normalized position of the argumentative element in the essay, and the frequency of significant bigrams in claims in the argumentative element are the three most important variables.

Table 4. Variable importance values

Variable	Importance Value
Word Count	289.988
Normalized element position in the essay	162.083
Frequency of significant bigrams in claims	47.992
Frequency of significant unigrams in claims	31.147
Normalized element position in the paragraph	29.791
Frequency of significant POS five grams in claims	28.465
Frequency of significant POS four grams in claims	27.389
Frequency of significant unigrams in non-claims	25.399
Frequency of significant POS unigrams in claims	25.272
Frequency of significant POS unigrams in non-claims	23.812
Frequency of significant POS trigrams in claims	20.364
Frequency of significant POS trigrams in non-claims	18.375
Frequency of significant POS four grams in non-claims	13.745
Frequency of significant four grams in claims	8.676
Frequency of significant POS six grams in claims	8.490
Frequency of significant POS five grams in non-claims	4.210

4. ERROR ANALYSES AND DISCUSSION

We conducted error analyses for the two Random Forest suprasentential models (i.e., the models based on the frequency and positionality feature set and the full feature set). Our goal was to examine the misclassifications of the models to better understand elements that may contribute to model accuracy.

We first examined classification rates. Among all incorrectly classified instances, we found more cases in which a claim was misclassified as a non-claim, whereas non-claims were less frequently misclassified as claims. For both models, around 17% of claims were misclassified and non-claims, and around 10% of non-claims were misclassified as claims. These results indicate that, the models are better at identifying non-claims than claims, potentially due to the imbalanced data between the claims and non-claims. Nevertheless, future studies should examine if there are more representative features in claims that can be integrated into our current feature set.

We next examined if essay quality and length influenced the model accuracy. Specifically, for each argumentative element in the two suprasentential test sets, we extracted the following information: holistic score, number of words, number of sentences, and number of paragraphs in the essay where the argumentative element occurred. We examined differences between the argumentative elements that were correctly and incorrectly predicted for these features using t-tests. No differences were reported for essay quality and length in either model. Thus, the classification of argumentative elements was not related to the quality or the length of essays.

We also examined if differences in model accuracy were related to more specific argumentation categories (i.e., micro-categories). As mentioned in Section 2.2, we merged the argumentation categories of Primary Claim, Final Claim, Counterclaim, and Rebuttal from the original annotated corpus into a larger classification of claims (i.e., a macro-classification). We also classified the remaining categories of Data and Concluding Summary along with Non-annotated texts into non-claims. To assess whether the micro-categories influenced classification of the macro-classification, we compared the prediction accuracies among the seven micro-categories.

The results showed that Counterclaims were not misclassified in either model (likely because of their rarity), Concluding Summaries were not misclassified in the frequency- and positionality-based models, but misclassified 3.9% of the time in the full feature model. Data was misclassified around 9% in both models. Meanwhile, the sub-categories that were more frequently misclassified included: Primary Claims (around 14 misclassified), Final Claims (around 21% misclassified), Non-annotated texts (around 22% misclassified), and Rebuttal (2 out of 3, 66.7% misclassified instances in both models). These results were also in line with findings that claims were more frequently misclassified as non-claims.

To further explore what factors affect the misclassifications among the micro-categories of argumentative types, Welch's t-tests were conducted among all NLP features (see Appendix B) used in the full analysis between correct and incorrect classification instances. However, the analysis was done for the sub-category of Counterclaim since all instances under this category were correctly predicted by the two models. Also, we did not conduct t-tests for the micro-category of Rebuttal due to a small sample size (N =3).

Table 5 presents the features for which significant differences were found between the correct and incorrect classification instances in at least two categories of argumentative types. In general, the classification of Primary Claim, Data, Concluding Summary, and Non-annotated texts seemed to be more strongly influenced by linguistic features. Word count was the strongest indicator of misclassification, in which difference were found for each micro-category. The standard deviation of dependents per object of prepositions was another strong predictor of misclassification, which reflects the development of syntactic complexity [25].

Table 5. Features with significant differences between correct and incorrect classification instances

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Primary claim		Yes	Yes		Yes	Yes	Yes			Yes		Yes	Yes	
Final claim		Yes									Yes			Yes
Data	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes			Yes	Yes	Yes	
Concluding summary	Yes	Yes						Yes	Yes		Yes		Yes	Yes
Nonannotated	Yes	Yes		Yes					Yes	Yes		Yes	Yes	

Note. Shaded gray cells with 'Yes' indicate significant difference ($p < .05$) were found between the correct and incorrect instances. 1 = Number of named entities, 2 = Word count, 3 = Normalized element position in the paragraph, 4 = Normalized element position in the essay, 5 = Frequency of significant unigrams in claims, 6 = Frequency of significant POS trigrams in claims, 7 = Frequency of significant quad-grams in claims, 8 = Hu Liu proportion score, 9 = Objects component score, 10 = Brown frequency score, 11 = Bigram lemma type-token ratio, 12 = Nouns as modifiers score, 13 = Dependents per object of the preposition (SD), 14 = T-units per sentence.

The number of named entities was a strong indicator for the non-claims, wherein the incorrect instances of non-claims contained fewer named entities versus the correct instances. The nouns as modifier scores were also predictive of misclassification, which measured the use of nouns as nominal modifiers in general and the variation in the number of modifiers per nominal [25]. Other linguistics features that influenced the classification accuracy included: the normalized position of the element in paragraph and in essay, the bigram type-token ratio, the frequency of key unigram, quad-gram, and POS trigram in claims, the number of T-units per sentence, the number terms that reference objects, the proportion of the number of words with positive sentiments to the words with negative sentiments, and the mean frequency score based on London-Lund Corpus of Conversation.

5. CONCLUSION

In this study, we proposed an approach that combined the frequency, positionality, and other lexical, syntactic, cohesion, and cognitive NLP features to predict claims and non-claims in argumentative essays. Our model performed well in the classification of these argumentative elements. Our exploration of the features, the comparison between sentential versus suprasentential models, and investigation of the factors that influenced classification accuracy in the error analyses should contribute to the field of automated identification and evaluation of discourse elements in argumentative writing.

It is important to note that the corpus used for this study was relatively small, comprising 314 student essays. Thus, to gain higher accuracies and reliabilities in classifying argumentative elements, we plan on annotating more essays and expanding the current corpus. That also means we will use essays written to more prompts allowing us to extract key n-grams and POS n-grams that are more generic and less restricted to the specific prompts used here. In addition, due to the small sample size, our classification of argumentative elements was simplified to focus on claims versus non-claims. We are interested in exploring the classification of the micro-categories (Primary Claim, Final Claim, Counter Claim, Rebuttal, Data, and Concluding Summary) in a larger corpus. We also plan to include the prediction of the quality of these argumentative elements in students' writing.

The models developed in this study will be included in an online Writing Assessment Tool (WAT). Implementing the classification algorithm within WAT, WAT's automatic writing evaluation (AWE) system will have the capacity to predict the number of claims in the essay and whether the claims mention the key n-grams that reflects the argument topic. This will afford providing feedback to students on argumentation quality within student essays. The study also provides insight into the length, position, content (e.g., the key n-grams), and other NLP features in claims versus non-claims in students' writing, which will contribute to finer-grained feedback components in our AWE system.

This study also provides important information for others who are developing AWE algorithms to drive feedback on argumentative essays, or more broadly to better understand the use of claims in essays. Specifically, the results of this study inform features related to feedback that can be provided to students about the number of claims, mentioning the argument topic, how to better position argumentative elements within their essays, and how to pay attention to specific linguistic features (such as the use of named entities when giving evidence) in their writing. This is an important achievement in the realm of writing feedback given the crucial need to automate feedback to students on their use of claims and evidence in argumentative essays.

Another important contribution of this study is that we also introduce a new corpus of essays annotated for argumentative elements, which is made publicly available at linguisticanalysistools.org. This corpus includes theoretically aligned argumentative elements that complement existing corpora [44, 45] and adds new components including prompts, holistic scores, additional categories of argumentation, and different educational settings. As such, this study provides the opportunity for other scientists to build upon our work such that we can better understand writing, and the features related to successful composition.

6. ACKNOWLEDGMENTS

The research reported here was supported by the Chan Zuckerberg Initiative, the Bill & Melinda Gates Foundation, and Schmidt Futures through grants to Georgia State University. Additional funding was provided by the Institute of Education Sciences, U.S. Department of Education, and the Office of Naval Research, through Grants R305A180261, R305A180144, N00014-20-1-2623, N00014-19-1-2424 to Arizona State University and University of New Hampshire. The opinions expressed are those of the authors and do not represent views of the funding agencies.

7. REFERENCES

[1] Aharoni, E., Polnarov, A., Lavee, T., Hershcovich, D., Levy, R., Rinott, R., Gutfreund, D. and Slonim, N., 2014, June. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the first workshop on argumentation mining* (pp. 64-68).

[2] Ajjour, Y., Alshomary, M., Wachsmuth, H. and Stein, B., 2019, November. Modeling frames in argumentation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 2915-2925).

[3] Al Khatib, K., Wachsmuth, H., Hagen, M., Köhler, J. and Stein, B., 2016, June. Cross-domain mining of argumentative text through distant supervision. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies* (pp. 1395-1404).

[4] Ashley, K.D. and Walker, V.R., 2013, November. From Information Retrieval (IR) to Argument Retrieval (AR) for Legal Cases: Report on a Baseline Study. In *JURIX* (pp. 29-38).

[5] Biber, D. and Conrad, S., 1999. Lexical bundles in conversation and academic prose. *Language and Computers*, *26*, pp.181-190.

[6] Bird, S., Klein, E. and Loper, E., 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[7] Burstein, J., Braden-Harder, L., Chodorow, M., Hua, S., Kaplan, B., Kukich, K., Lu, C., Nolan, J., Rock, D. and Wolff, S., 1998. Computer analysis of essay content for automated score prediction: A prototype automated scoring system for GMAT analytical writing assessment essays. *ETS Research Report Series*, *1998*(1), pp.i-67.

[8] Burstein, J., Kukich, K., Wolff, S., Lu, C. and Chodorow, M., 2001a. Enriching Automated Essay Scoring Using Discourse Marking.

[9] Burstein, J., Marcu, D. and Knight, K., 2003. Finding the WRITE stuff: Automatic identification of discourse structure in student essays. *IEEE Intelligent Systems*, *18*(1), pp.32-39.

[10] Burstein, J., Marcu, D., Andreyev, S. and Chodorow, M., 2001b, July. Towards automatic classification of discourse elements in essays. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics* (pp. 98-105).

[11] Cohen, J., 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, *20*(1), pp.37-46.

[12] Cohen, R., 1984, July. A computational theory of the function of clue words in argument understanding. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics* (pp. 251-258).

[13] Core Team, R., 2017. R: A language and environment for statistical computing. R Foundation for Statistical Computing. *Vienna, Austria: URL https://www. R-project. org/.[Google Scholar]*.

[14] Crossley, S.A., Kyle, K. and McNamara, D.S., 2016. The development and use of cohesive devices in L2 writing and their relations to judgments of essay quality. *Journal of Second Language Writing*, *32*, pp.1-16.

[15] Crossley, S.A., Kyle, K. and McNamara, D.S., 2017. Sentiment Analysis and Social Cognition Engine (SEANCE): An automatic tool for sentiment, social cognition, and social-order analysis. *Behavior research methods*, *49*(3), pp.803-821.

[16] Fleiss, J.L., Levin, B. and Paik, M.C., 1981. The measurement of interrater agreement. *Statistical methods for rates and proportions*, *2*(212-236), pp.22-23.

[17] Fraser, B., 1999. What are discourse markers?. *Journal of pragmatics*, *31*(7), pp.931-952.

[18] Grosz, B. and Sidner, C.L., 1986. Attention, intentions, and the structure of discourse. *Computational linguistics*.

[19] Hirschberg, J. and Litman, D., 1993. Empirical studies on the disambiguation of cue phrases. *Computational linguistics*, *19*(3), pp.501-530.

[20] Houngbo, H. and Mercer, R.E., 2014, June. An automated method to build a corpus of rhetorically-classified sentences in biomedical texts. In *Proceedings of the first workshop on argumentation mining* (pp. 19-23).

[21] Kilgarriff, A., 2001. Comparing corpora. *International journal of corpus linguistics*, *6*(1), pp.97-133.

[22] Knott, A. and Dale, R., 1994. Using linguistic phenomena to motivate a set of coherence relations. *Discourse processes*, *18*(1), pp.35-62.

[23] Kuhn, M., 2015. A Short Introduction to the caret Package. *R Found Stat Comput*, *1*.

[24] Kyle, K. and Crossley, S.A., 2015. Automatically assessing lexical sophistication: Indices, tools, findings, and application. *Tesol Quarterly*, *49*(4), pp.757-786.

[25] Kyle, K., 2016. Measuring syntactic development in L2 writing: Fine grained indices of syntactic complexity and usage-based indices of syntactic sophistication.

[26] Lawrence, J. and Reed, C., 2015, June. Combining argument mining techniques. In *Proceedings of the 2nd Workshop on Argumentation Mining* (pp. 127-136).

[27] Levy, R., Bilu, Y., Hershcovich, D., Aharoni, E. and Slonim, N., 2014, August. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (pp. 1489-1500).

[28] Liaw, A. and Wiener, M., 2002. Classification and regression by randomForest. *R news*, *2*(3), pp.18-22.

[29] Lippi, M. and Torroni, P., 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, *16*(2), pp.1-25.

[30] Liu, F. and Stapleton, P., 2014. Counterargumentation and the cultivation of critical thinking in argumentative writing: Investigating washback from a high-stakes test. *System*, *45*, pp.117-128.

[31] Max, K., 2016. Contributions from Jed Wing. *Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team*,

Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang and Can Candan, pp.6-0.

[32] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.C. and Lin, C.C., 2015. Misc functions of the department of statistics, probability theory group (formerly: E1071). *Package e1071. TU Wien.*

[33] Nguyen, H. and Litman, D., 2015, June. Extracting argument and domain words for identifying argument components in texts. In *Proceedings of the 2nd Workshop on Argumentation Mining* (pp. 22-28).

[34] Nguyen, H. and Litman, D., 2016, August. Context-aware argumentative relation mining. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1127-1137).

[35] Nussbaum, E.M., Kardash, C.M. and Graham, S.E., 2005. The Effects of Goal Instructions and Text on the Generation of Counterarguments During Writing. *Journal of Educational Psychology*, *97*(2), p.157.

[36] Ong, N., Litman, D. and Brusilovsky, A., 2014, June. Ontology-based argument mining and automatic essay scoring. In *Proceedings of the First Workshop on Argumentation Mining* (pp. 24-28).

[37] Palau, R.M. and Moens, M.F., 2009, June. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law* (pp. 98-107).

[38] Persing, I. and Ng, V., 2015, July. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 543-552).

[39] Quirk, R., 1985. The English language in a global context. *English in the world: Teaching and learning the language and literatures*, *16*, pp.17-21.

[40] Rayson, P. and Garside, R., 2000, October. Comparing corpora using frequency profiling. In *The workshop on comparing corpora* (pp. 1-6).

[41] Rinott, R., Dankin, L., Alzate, C., Khapra, M.M., Aharoni, E. and Slonim, N., 2015, September. Show me your evidence-an automatic method for context dependent evidence detection. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 440-450).

[42] Schiffrin, D., 2001. Discourse markers: Language, meaning, and context. *The handbook of discourse analysis*, *1*, pp.54-75.

[43] Song, Y., Heilman, M., Klebanov, B.B. and Deane, P., 2014, June. Applying argumentation schemes for essay scoring. In *Proceedings of the First Workshop on Argumentation Mining* (pp. 69-78).

[44] Stab, C. and Gurevych, I., 2014, October. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 46-56).

[45] Stab, C. and Gurevych, I., 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, *43*(3), pp.619-659.

[46] Toulmin, S.E., 2003. *The uses of argument*. Cambridge university press.

[47] Van Eemeren, F.H., Houtlosser, P. and Henkemans, A.F.S., 2008. Dialectical profiles and indicators of argumentative moves. *Journal of Pragmatics*, *40*(3), pp.475-493.

[48] Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L.D.A., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J. and Kuhn, M., 2019. Welcome to the Tidyverse. *Journal of Open Source Software*, *4*(43), p.1686.

APPENDIX

A. Definitions of argumentative elements

Elements	Definitions	Examples
Final Claim	An opinion or conclusion on the main question	In my opinion, every individual has an obligation to think seriously about important matters, although this might be difficult.
Primary Claim	A claim that supports the final claim.	The next reason why I agree that every individual has an obligation to think seriously about important matters is that this simple task can help each person get ahead in life and be successful.
Counterclaim	A claim that refutes another claim or gives an opposing reason to the final claim.	Some may argue that obligating every individual to think seriously is not necessary and even annoying as some people may choose to just follow the great thinkers of the nation.
Rebuttal	A claim that refutes a counterclaim.	Even though people can follow others' steps without thinking seriously in some situations, the ability to think critically for themselves is a very important survival skill.
Data	Ideas or examples that support primary claims, counterclaims, or rebuttals.	For instance, the presidential debate is currently going on. In order to choose the right candidate, voters need to research all sides of both candidates and think seriously to make a wise decision for the good of the whole nation.
Concluding Summary	A concluding statement that restates the claims.	To sum up, thinking seriously is important in making decisions because each decision has an outcome that affects lives. It is also important because if you think seriously it can help you succeed.
Non-annotated	Any text that doesn't fall into any of the above categories	People always strive to be unique or different. This idea clashes with creativeness all through our lives.

B. Descriptions of the SALAT NLP features

NLP features from SALAT	Descriptions
Bigram lemma type-token ratio	Number of unique bigram lemmas (types) divided by the number of total bigram lemmas (tokens)
Brown frequency score	Mean word frequency score based on London-Lund Corpus of Conversation
Brysabaert concreteness score	Sum of concreteness scores based on all words divided by number of words with concreteness scores
COCA academic bigram association strength	Sum of approximate collexeme strength score divided by the number of bigrams in text with collexeme scores
Dependents per clause (SD)	The standard deviation of the total number of dependents per clause
Dependents per object of the preposition (SD)	This score captures the variation (standard deviation) in the prepositional objects
Direct objects per clause	The number of direct objects per clause
Free association tokens response score	Number of response tokens elicited by word as stimuli in discrete word association experiment (based on function words)
Hu Liu proportion score	Proportion of the number of words with positive sentiments to the words with negative sentiments
LDA age of exposure score	Based on Incremental Age of Exposure for words across 13 grade levels; calculated based on 1/slope of linear regression
Lexical decision time	Standardized lexical decision reaction time across all participants for this word (z-score, based on function words)
Nouns as modifiers score	This score captures the use of nouns as modifiers and modifier variation
Number of named entities	The number of named entities
Number of prepositions per clause	This score captures capture noun phrase elaboration and clause complexity
Objects component score	This component score represents the number of terms that reference objects
Possessives component score	This component score captures the use of possessives in general, and specifically captures the use of possessives in nominal subjects, direct objects, and prepositional objects
Sentiment score of dominance	This score captures the sentiment of dominance, measured by the number of words of dominance
Sentiment score of overstating	This score captures the sentiment of overstating, calculated based on words indicating emphasis in realms of frequency, causality, accuracy, validity...
T-units per sentence	Number of T-units in text divided by number of sentences in text
Verb argument constructions association strength	Average approximate collostructional strength score based on the COCA academic corpus

Note. For more information about the SALAT NLP features, please see https://www.linguisticanalysistools.org/

Using Keystroke Analytics to Understand Cognitive Processes during Writing

Mo Zhang, Hongwen Guo, and Xiang Liu
Educational Testing Service (Princeton, NJ, 08541)
MZhang, HGuo, XLiu003@ETS.org

ABSTRACT

We present an empirical study on the use of keystroke analytics to capture and understand how writers manage their time and make inferences on how they allocate their cognitive resources during essay writing. The results suggest three distinct longitudinal patterns of writing process that describe how writers approach an essay task in a writing assessment. Discussion of the potential applications of keystroke analytics for improving teaching, learning, and assessing writing are also provided.

Keywords

keystroke logging, writing, cognitive process, time management, writing pattern

1. INTRODUCTION

The study of writing process has long been of interest to the writing research community (e.g., [4], [18], [12], [19], [22]). With the advances in technology, keystroke logging has become a practical and popular tool to capture and study the process of composition in a wide range of contexts [10]. In this study, we demonstrate some research findings on the use of keystroke analytics to understand writers' time management during their writing process. The results have practical implications for the teaching and learning of writing in classrooms.

Previous research on writing cognition suggested several subprocesses of writing [9], including task analysis, text planning, idea generation, translating ideas into natural language, transcribing langauge onto paper (handwriting) or a screen (keyboard-based writing), text revision, copy editing and reviewing. Figure 1 illustrates a simplified version of Hayes coginitive writing model, which specified four main subprocesses of writing. Specifically, idea generation and task preparation (i.e., *proposer*) often manifest as pauses at the start of writing and at sentence boundaries; fluency of putting ideas into language (i.e., *translator*) primarily relates to the size of long sequences of text production without major interruption (also known as "burst"); orthographic proficiency and motor skill (i.e., *transcriber*) typically relates to pauses inside a word and to edits designed to make immediate corrections to spelling errors or typos; and editing and reviewing (i.e., *evaluator*) usually show up as jumps to different locations in the text to make changes or replace large chunks of existing text with new content.

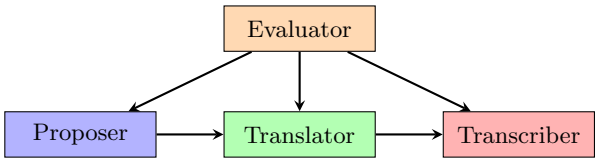


Figure 1: A Cognitive Model of Writing Process

One important implication from the cognitive model is that writing is not a linear process and successful writing calls for effecitive management and coordiation of the subprocesses. Drawing from the cognitive theories of writing, an overview of the types of activities occuring during text composition can be found in [5]. The cognitive resources, as stated in [3], required to carry out each activity do not distribute randomly over the text-production process. Writers often need to decide on which goals to prioritize at which time point because they simply do not have unlimited working memory to accomplish everything at once [11]. With the availability of keystroke logs, how writers distribute their time and cognitive resource to various subprocesses of writing can be quantified and analyzed, which is described in the next section. In this study, we aim to tackle a specific *research question* of whether there are distinct writing-process patterns that may be detected with regard to how writers allocate their cognitive resource to various subprocesses of writing. An identification of meaningful writing profiles will have practical implications for instructors to design curriculum suitable to their classes, and personalize their instruction for learners with different needs and characteristics.

1.1 Keystroke Logging

We consider keystroke logging as a recording of every keypress that the writer makes on the keyboard. The gap time between two consecutive keypresses is often called an interkey interval (IKI), which is also recorded in keystroke logging. A single keystroke record in JSON format may look like this: {"p": "1", "o": "", "n": "I", "t": "0.57"}, where

Mo Zhang, Hongwen Guo and Xiang Liu "Using Keystroke Analytics to Understand Cognitive Processes during Writing". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 384-390. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Table 1: An Example Keystroke Log Segment

Index	PosInText	Content	ContentLen	TimeStamp	ActionType	Context	WordIntended	CursorJump	TextToDate
0	1	I	1	0.57	Insert	WordStart	It	N	I
1	2	t	1	0.62	Insert	InWord	It	N	It
2	3	(*space*)	1	0.99	Insert	BetweenWords		N	It
3	4	i	1	1.30	Insert	WordStart	is	N	It i
4	5	s	1	1.42	Insert	InWord	is	N	It is
5	6	(*space*)	1	1.61	Insert	BetweenWords		N	It is
6	7	a	1	2.12	Insert	WordStart	a	N	It is a
7	8	(*space*)	1	2.22	Insert	BetweenWords		N	It is a
8	9	g	1	2.35	Insert	WordStart	fun	N	It is a g
9	10	o	1	2.43	Insert	InWord	fun	N	It is a go
10	11	o	1	2.55	Insert	InWord	fun	N	It is a goo
11	12	d	1	2.68	Insert	InWord	fun	N	It is a good
12	12	d	1	3.01	Delete	InWord	fun	N	It is a goo
13	11	o	1	3.16	Delete	InWord	fun	N	It is a go
14	10	o	1	3.30	Delete	InWord	fun	N	It is a g
15	9	g	1	3.53	Delete	InWord	fun	N	It is a
16	10	f	1	3.71	Insert	InWord	fun	N	It is a f
17	11	u	1	3.93	Insert	InWord	fun	N	It is a fu
18	12	n	1	4.11	Insert	InWord	fun	N	It is a fun
19	13	(*space*)	1	4.30	Insert	BetweenWords		N	It is a fun
20	14	d	1	4.49	Insert	WordStart	day	N	It is a fun d
21	15	a	1	4.62	Insert	InWord	day	N	It is a fun da
22	16	y	1	4.90	Insert	InWord	day	N	It is a fun day
23	17	.	1	5.13	Insert	PuncMark		N	It is a fun day.

Note: ContentLen=Content Length. ContentLen usually takes the value of 1 unless the writer cuts or pastes in a chunk of text with more than one character. PuncMark=Punctuation Mark. CusurJump can take a binary value of "Y" or "N".

"p" is the position in the text box, "o" is the current text at that position, "n" is the change made to that position, and "t" is the time elapsed since the start of writing. In this example, the writer inserted a chatacter "I" at position 1 in the text box at a timestamp of 0.57 seconds, computed relative to when the writing started (i.e., at 0 elapsed seconds). The overall behaviral process of text production can then presented by a sequence of keystroke records. More importantly, qualitative labels may be attached to characterize a keystroke record in terms of the type (e.g., insertation, deletion) and location (e.g., inside of a word, end of a sentence) of an action, along with the content and associated time stamp. For the hypothetical example given in Table 1 (for illutration purpose only), the writer spends 5.13 seconds to write a full sentence: "It is a fun day." During the process, the writer changed the choice of a word from "good" to "fun" evidenced by a sequence of the "delete" actions. Cursor location is tracked so that if the cursor moves suddenly to a different location away from the current location, the jump behavior can be detected.

As Table 1 indicates, keystroke logs allow the visible aspects of the text-production process to be precisely reconstructed and retrospectively replayed. Figures 2 and 3 demonstrate one approach to visualizing the dynamics of the text-production process by plotting the time elapsed (horizontal axis) against text length and cursor position (verticle axis). When the writer is appending or deleting text at the end of the text, the dashed purple line (text length) and the solid green line (cursor position) would converge; when the writer is making changes elsewhere in the text, the green cursor-line would diverge from the purple length-line. The gaps between the two lines can indicate the degree of the "jump" action. The length-line can go up or down indicating adding or removing of content. The small-scale zig-zag pattern in both figures suggests that both writers conducted a fair amount of quick fixes or local edits mostly on the word level (e.g., typo correction, word-choice revision, removing/adding punctuation marks) at the end of the text as they write. The writer in Figure 3 showed evidence of global-level editing behavior towards the end of the writing session, when the writer moved the cursor to different parts of the text to make changes, as can be seen from the relatively large gaps between the purple and green lines. This type of jump-edit behavior is rather absent in Figure 2 for which the writer showed a much more linear writing process.

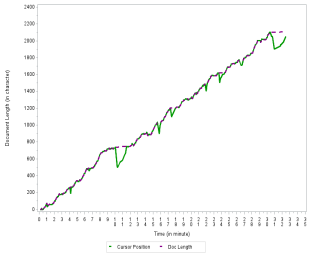


Figure 2: Writing Progression Example a

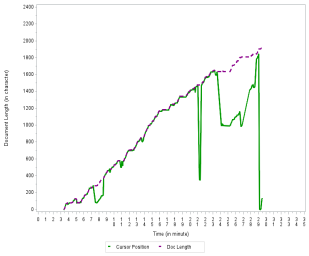


Figure 3: Writing Progression Example b

1.2 Inferences & Relations to Writing Quality

The nature and location of the changes that writers make to their text directly can directly support inferences about where the writer is cognitively in the composition process. For example, a long pause followed by insertion of a written outline is suggestive of task analysis and idea generation; a long pause at the phrasal or sentence boundaries followed by a burst of text production is a sign of sentence planning; alternating between insert and delete actions on a character-level inside of a word is likely an indication of spelling correction or word finding; if a writer types long sequences of words thereby adding new content, it is reasonably safe to assume that the writer is primarily engaged in content generation; and, if a writer jumps to various locations (tracked by cursor position) in the text to make changes, the writer is more likely in the state of text reviewing and revision (e.g., [6], [16]). Previous research has reported that the

process of writing, such as the total time spent on writing, between-word pause tempo, initial pause length before typing a word, length of long burst (i.e., stretches of long sequence of text production), and extent of text editing and revision, relate to the quality of the final written product (e.g., [3], [17], [2], [27]). In this study, we largely followed the practice described in [8] by classifying each interkey interval (i.e., gap time betweeb two keystrokes) into one of the following four cognitive states in writing. These states intend to operationalize the theoretical subprocesses proposed in Hayes model, although there will unavoidably be gaps between theory and practice. *Long Pause* state, representing text planning, idea generation and deliberation, or hesitation and struggle with text production; *Text Production* state, representing relatively fluent content generation without major interruption where interruption is signaled by an extended pause; *Local Editing* state, representing localized (mostly on the word-level) minor text editing; and *Global Editing* state, representing reviewing, revision and copy editing on the whole passage/text level.

2. METHOD

2.1 Data Set

The data set was collected from a high-school equivalency testing program, which contains five subject tests: English language arts – reading, English language arts – writing, math, science, and social science. The focus of this study was the essay writing task in the writing subtest. In responding to the essay task, the examinees are expected to read two sources with different perspectives on a common issue (e.g., whether success is more the result of talent or of hard work), and then express and explain their opinions in writing while appropriately incorporating evidence from the sources. Each submitted essay was scored holistically on a 0-6 scale by two trained human raters according to a standardized grading rubric. Essays receiving a human score of 0 were excluded from analysis as those essays tend to have aberrant characteristics such as being empty, not in English, or consisting of random keystrokes. In this study, we selected two writing forms administered between September 2017 and August 2018 for investigation. Each form contained one essay writing task, or prompt. The sample size used for analysis was approximately 500 in each prompt. The analyses were conducted on the first (base) prompt, and then replicated on the second prompt to validate the consistency of the findings.

2.2 Propensity Score Matching

To ensure comparability, propensity score matching (PSM) [1] was used to minimize irrelevant factors such as performance level and the paraticipants' demographics and to balance covariates between the participants who responded to either of the two prompts. Also, the two different prompts were not administered at random, thus necessitating this step. A logistic regression model was developed to generate the propensity scores, and a one-to-one greedy matching without replacement algorithm with a caliper value of 0.05 was applied to find the matches in the prompt 2 sample to make it most comparable to the prompt 1 sample [15]. The caliper value refers to the maximum distance in propensity scores; hence the smaller the caliper value, the closer the match. In performing the PSM, the covariates were chosen based on our understanding of the examination and the examinee population, and on findings from previous reports on subgroup differences in writing process (e.g., [7], [25]). The covariates included for propensity score matching were gender (Male or Female), ethnicity (White, Black, Hispanic, or others/unreported), employment status (Full-time, Part-time, Unemployed, or others/unreported), highest education level (Below Grade 9, Some high school, others/unreported), English as best communicative language (Yes or No), as well as scores on the subject tests other than writing. All the demographic background variables were self-reported by the participants on a voluntary basis.

2.3 Feature Extraction

Keystroke logs were recorded automatically as writers composed their essays. A two-stage procedure was applied for feature extraction. In Stage 1, we classified each interkey interval (IKI) into one of four heuristically-defined and mutually-exclusive writing states by following the practice in [8] with some modifications. In Stage 2, we split each log into ten time periods by evenly dividing the total writing duration into ten segments, as one way to align and compare the logs of different length in duration. The choice of ten time-periods was made to balance the duration of each segment, which should be long enough to detect any patterns related to time distribution, *and* the number of segments, which should be sufficient for detecting longitudinal patterns.

Stage 1: Classification of writing states. The general programming logic is as follows.

- Step 1. Define Long Pause (LP) state. If an IKI is longer than n times in-word typing speed, it is then labelled as P. The keystroke sequence in between two adjacent Ps is considered a burst.
- Step2: Define Text Production (TP) state. Inside of a burst, if there is an absence of Delete action, or the max number of a Delete sequence is smaller than k, label all IKIs in this burst as TP. If there is a consecutive delete action sequence with k or more number of Deletes, temporarily change all IKIs in this burst to R, which will be refined in the next step.
- Step 3. Define Local Editing (LE) state. Use an m-IKI moving window to scan through the keystroke sequence within an R-burst. That is, the first moving window contains the 1^{st} to the m^{th} IKIs in a burst; the second moving window contains the 2^{nd} to the $(m+1)^{\text{th}}$ IKIs in the burst; and so on.
  - If all IKIs in a moving window are Inserts or contain less than s Deletes, change the first record in the moving window from R to TP. Continue with the same logic to the next moving window.
  - If a moving window contains equal to or more than s Delete actions, label all the m records in the moving window as LE.
- Step 4. Define Global Editing (GE) state. GE is indicated by text deletion while crossing sentence boundaries or making jump-edits elsewhere in the text away from the current location.

- If d or more consecutive Delete actions contain the following punctuation marks – comma, period, semicolon, exclamation mark, and question mark – label the entire Delete sequence as GE.
- If the position of an IKI in the text is different from the one before it by more than q, then the IKI is labeled as Jump Back (JB to an earlier place in the text) or Jump Forward (JF to a later place). Find the longest JB-JF pair in distance, and label all the IKIs in between to GE.

The parameters n, m, k, s, d, and q used in state definitions are customizable. In this study, we chose n=10, m=3, k=3, s=3, d=2, and q=4 as a starting point, mainly following [8]. Once all the IKIs are labelled, consecutive IKIs of the same state can be further aggregated. Each keystroke log can be described by the total number of states, the duration of each state, proportion of time spent on each state, the frequency of various transitions, etc. Among the four states, there are 12 state-transition possibilities (e.g., TP $\rightarrow$ GE) in total.

Stage 2: In this stage, we used the state classification generated above to calculate a writer's time distribution at various points during the writing process. To accomplish this goal, we divided each keystroke log into ten even time-periods. We then calculated the proportion of time spent in each writing state within a time period. As a result, each keystroke log was represented by a vector of 40 elements (i.e., four states times ten segments). The elements' values (i.e., percentages) could range from 0 to 100. When it is a 0, it simply means that the writer did not spend time on an activity (e.g., Local Editing) during that time period (which is one tenth of the total time). Similarly, when the value is 100, it means that a writer spent all his/her time on an activity (e.g., Global Editing) during that time period. With this information, the longitudinal pattern of time allocation can be revealed and investigated. We can analyze, for example, how writers spend their time at the beginning, middle, or end of their writing process, whether writers distribute their effort evenly or differently at various time-points during the writing process, and whether there are distinct profiles with regard to time management of an individual writing process.

2.4 Cluster Analysis and Interpretation

Using the writing samples in each of the two prompts, we conducted hierarchical cluster analysis (agglomerative approach) with Ward linkage using the Euclidean distance metric [23, 20]. The proportion of time spent in each state at the ten time-points was used to create 40 input variables. Each input variable was standardized to a mean of zero and standard deviation of one across individuals in a prompt [13]. The cluster analysis was done separately for each prompt, with the second prompt serving as a replication sample to verify results from the base prompt. Because there is no established convention for choosing the number of clusters, we used the Pseudo-F statistic, model R-squared, and semipartial R-square statistics to help us determine the appropriate number. The pseudo-F statistic is calculated as the ratio of the between-cluster variance to the within-cluster variance [14]. Larger values indicate better separation between the clusters. The model R-squared indicates the proportion of variance accounted for by the clusters. The semipartial R-square indicates the decrease in the proportion of variance accounted for due to joining two clusters. We plotted these statistics against the number of clusters to examine the impacts of joining or splitting clusters. Dendrograms visualizing the distances between the keystroke logs were also examined to help select the final number of clusters.

To interpret identified clusters, we compared how writers falling into different clusters allocated their efforts during the writing process. Since the proportions of time spent on Text Production, Local Editing, Global Editing, and Long Pause were used as input variables in the cluster analysis, this comparison would be the most direct way to examine any distinct patterns exhibited by each cluster. It would also be informative to know whether clusters are associated with distinct patterns of writing proficiency or in cluster members' demographic background. Therefore, to further substantiate the meaning of identified clusters, we compared the essay scores, essay length (in words), time on task (in minutes), the proportion of cluster members belonging to specific demographic categories, as well as a rough measure of writing efficiency calculated as essay length divided by time on task, between the clusters.

3. RESULTS

3.1 Outcome of Propensity Scoring Matching

The samples resulting from the propensity score matching are closely comparable between the two prompts (Table 2). In the first (base) prompt, males and females were evenly distributed; 53% of the examinees self-identified as White, 12% as Black, 14% as Hispanic; 4% reported that their highest grade level was below Grade 9, 62% reported having some high school education; 16% were working part-time, 17% were working full time, 23% were unemployed at the time of the examination. The majority of the examinees, 94%, indicated English as their best communicative language. The demographic background distribution of the second prompt, after matching, was very similar to that for the base prompt. The matched samples for prompt 2 also showed comparable means of the subtest scores to those for the base prompt.

3.2 Cluster Analysis Results

To decide the optimal number of clusters, we first examined the dendrograms, which indicated a solution of 3 or 4 clusters for both prompts. Figures 4 and 5 show the results of the Pseudo-F statistic and Sempartial R-square statistic that were considered for model selection in prompt 2. The results for prompt 1 are similar. The X-axis in both plots is the number of clusters ranging from 1 to 50. The Y axis is the Pseudo-F statistic on the left plot, R-squared on the middle plot, and semipartial R-square on the right plot. The results suggested a peak on the Pseudo-F statistic with a 3-cluster solution. The semipartial R-square plot shows an elbow point at cluster 3. The model R-squared (not shown) appears to go up continuously without a clear turning point. Considering all the evidence, we decided on three clusters as the most parsimonious and sensible solution for the this study sample.

3.3 Comparing the Clusters

Given the multivariate nature of the clustering variables, we drew radar charts to visualize the time distribution at the

Table 2: Comparability between Prompts after Propensity Score Matching

	Proportion										Mean Score (Scale: 0 - 20)			
Prompt	Female	White	Black	Hisp.	Below9	HS	PT	FT	Unemp.	Eng(Y)	Reading	Math	Science	SS
1 (base)	0.50	0.53	0.12	0.14	0.04	0.62	0.16	0.17	0.23	0.94	12.23	11.55	14.11	13.41
2 (matched)	0.51	0.52	0.13	0.14	0.02	0.63	0.16	0.19	0.20	0.94	12.19	11.73	13.97	13.55

Note: Below9: Below Grade 9; HS: some high school; PT: part-time employee; FT: full-time employee; Unemp.: unemployed; Eng(Y): English as best communicative language; SS: Social Science

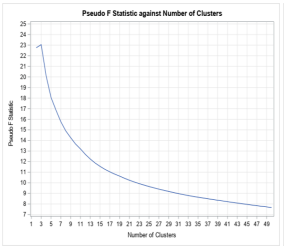


Figure 4: Pseudo-F statistic x N of Cluster

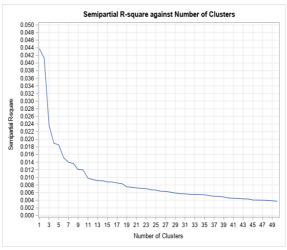


Figure 5: Semipartial R^2 x N of Cluster

ten consecutive time-points during the writing process of the logs belonging to each of the three clusters (Figures 6 and 7). The axes represent the average proportions of time spent on a state at a certain time. For example, "TP_1" refers to the proportion of time spent on Text Production at the beginning of writing – the 1st of the ten duration segments. It is clear from Figures 6 and 7 that the three clusters demonstrated rather different polygonal shapes over all axes, which are consistent between the two prompts.

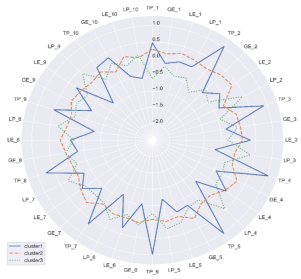


Figure 6: Radar Chart (Prompt 1)

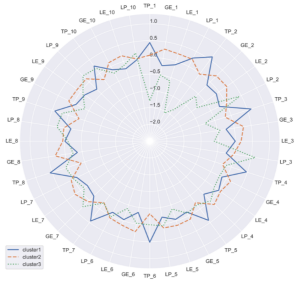


Figure 7: Radar Chart (Prompt 2)

Cluster 1 (blue colored, solid line in both prompts) writers have a distinct pattern with notable spikes on the TP state over the course the writing process. Because the total writing time is constrained and writers can only do one thing at a time, if the writers spend more time on text production, they would necessarily spend less time on the other activities such as editing or revision. This constraint is evident from the plots where, for Cluster 1, the proportion of time spent on long pauses and editing is relatively smaller. Cluster 2 (orange colored, dashed line) writers, on average, have a much smoother circle compared to Cluster-1. Cluster 2 writers appeared to have distributed their efforts more evenly throughout the writing process. The allocation of time across the four writing states is relatively balanced over the course of the writing session. In general, Cluster 1 seems to represent a group of writers that compose linearly without showing much editing behaviors, while Cluster 2 seems to represent writers that also consistently produce text but still spend time on text planning and conduct text editing and revision as they write. Cluster 3 (green colored, dotted line) writers further showed a distinct time-management pattern from the other two clusters. The writers in Cluster-3 appeared to have difficulties in generating text at the start of the writing session, as evidenced by the lack of text production (TP_1) and a higher proportion of the local editing behaviors (GE_1 and LE_1) in the first time period, which suggested possible false starts during the writing process. This "struggling" pattern appeared to have persisted into later stages of the writing process, as evidenced by a higher proprotion of time spent on long pauses compared to text production or editing.

We also examined the actual length of time writers stayed in a state before transitioning to a different state. The results suggest that the three clusters not only differ in their relative time distributions during writing, but also in the total time writers stayed in different states. Each cluster displayed distinct patterns consistent across prompts: Cluster 1 writers spent considerably less time on long pauses than Clusters 2 and 3 writers; Cluster 2 writers spent notably larger amounts of time making word-level local edits than Clusters 1 and 3 writers by approximately 1 minute in Prompt 1 and 2 minutes in Prompt 2; and Clusters 3 writers generally spent less time on text production than Clusters 1 and 2 writers by about 1-2 minutes in Prompt 1 and about 3.5 minutes in Prompt 2. An additional interesting difference between Clusters 1 and 2 is that Cluster 2 writers not only spent longer time on local editing, but also on global editing by about 3 minutes in both prompts. A close comparison between Clusters 2 and 3 further revealed that Cluster 2 writers also appeared to spend more time on long pauses than Cluster 3 writers by a small margin in Prompt 1 and by a rather large margin in Prompt 2.

Taking into account both absolute time spent in each state, and relative time in each state, Cluster 2 writers appear to have shown a stable-tempo, iteractive process pattern in which they switch repeatedly between the activities of text planning, text production, and text editing over the course of their writing sessions. Although more data are needed to verify this interpretation, the long pauses demonstrated by Cluster 3 writers throughout the writing session appeared to be signals of hesitation and difficulties in content generation. Finally, Cluster 1 writers seem to be relatively quick and fluent at generating ideas (as evidenced by fewer long pauses) and at translation and transcription (that is, at expressing their ideas in written form).

To better understand and interpret the identified clusters, Table 3 further compares the characteristics of student essays across the three clusters. Cluster 1 writers spent the shortest time on the writing task on average (22.94 minutes

Table 3: Proficiency and Demographic Distributions of Clusters

		Prompt 1			Prompt 2		
	Variable	Cluster 1	Cluster 2	Cluster 3	Cluster 1	Cluster 2	Cluster 3
	n	65	264	168	169	259	71
	Essay Score (scale: 2 - 12)	5.51 (1.70)	5.64 (1.54)	5.43 (1.53)	5.38 (1.59)	5.42 (1.74)	4.51 (1.80)
	Essay Length (in words)	305 (136)	324 (145)	289 (128)	287 (129)	290 (127)	236 (118)
	Time on Task (in minutes)	22.94 (11.56)	36.60 (15.65)	32.36 (14.16)	25.45 (12.39)	36.04 (14.86)	26.88 (14.37)
	Efficiency (words/min)	15.02 (6.19)	9.94 (4.43)	9.96 (4.09)	12.85 (5.71)	8.97 (3.98)	10.82 (6.41)
	Female	0.52	0.56	0.43	0.49	0.54	0.39
	White	0.62	0.51	0.50	0.58	0.51	0.46
	Black	0.12	0.14	0.13	0.12	0.12	0.13
	Hispanic	0.06	0.13	0.18	0.09	0.17	0.13
Proportion	Below Grade 9	0.03	0.02	0.02	0.05	0.03	0.03
	Some high school	0.54	0.63	0.64	0.60	0.62	0.68
	Part-time	0.15	0.13	0.20	0.17	0.16	0.14
	Full-time	0.12	0.19	0.22	0.17	0.17	0.20
	Unemployed	0.20	0.21	0.18	0.21	0.24	0.28
	English as Best (Y)	0.96	0.94	0.96	0.96	0.92	0.94

Note: Values in parenthesis are standard deviations. Values on the lower-half of the table are the percent of various subgroups within a cluster.

in Prompt 1; 25.45 minutes in Prompt 2). The difference in the total time on task between Clusters 1 and 2 is drastic – about 14 minutes difference in Prompt 1 and about 10 minutes difference in Prompt 2. Cluster 1 writers wrote notably more words/minute than Cluster 2 writers (15.02 vs 9.94 in Prompt 1; 12.85 vs. 8.97 in Prompt 2). The overall evidence seems to suggest Cluster 1 writers were more efficient than Cluster 2 writers, in that they spent significantly less time writing, yet achieved comparable text quality (essay scores).

In relation to demographic background, several results are noteworthy. On both prompts, Cluster 1 contained a notably greater proportion of White writers, a lower proportion of Hispanic writers, and a lower proportion of examinees with high-school experience, compared to the overall demographic distribution in Table 2. Cluster 2 included a slightly greater proportion of female writers than the average on both prompts, while all other demographic variables fell close to the mean for each prompt. Finally, Cluster 3 had a considerably lower proportion of White or female writers. But the results in general are less consistent between the two prompts for Cluster 3. The evidence seems to suggest that writers from different demographic background and having different educational experience may display distrinctive patterns in their writing processes. However, without further evidence, we cannot infer any causal connection between demographic group membership, writing process patterns, and overall writing performance.

4. DISCUSSION

In this paper, we presented a study on the use of keystroke analytics to understand writers' cognitive processes during writing. One possible outcome of this study, and of the larger research program of which it is a part, would be to providing actionable writing feedback to instructors and learners. However, before we can reach this goal, we need to develop a clear understanding of how writing processes change as a result of learning and instruction. This study provides a first attempt to address this issue, by identifying characteristic longitudinal patterns of time management that writers display when they respond to an essay writing task. The current results suggest that there are at least three distinct writing profiles that describe how writers approach an on-demand essay writing task. Though, it will be critical that the analysis to be replicated with more data. In the future, for writers placed into different profiles, we can imagine giving customized suggestions on writing strategies to improve learning and practice. Obviously, more research is needed to further validate the meaning and interpretation of the profiles we have detected. Possible approaches include cognitive interviews to elicit writers' understanding of what they were doing, combining eye-tracking technique with keystroke logging to get a better sense of where the writer's attention was focused and how changes in the focus of attention interacts with pause patterns, and convening an expert panel to determine whether the clusters derived by atatistical analysis align with expert judgments about what the writers were doing at each point in the writing process. The availability of keystroke logs makes it possible to replay a writer's composition process like a movie. Such replays can then be presented as stimuli to assist and guide cognitive interviews or the expert review process. Statistical tests will be necessary to detect if the profiles are significantly different beyond the practical importance. It will also be essential to replicate our analyses on a wide range of writing prompts, a broader variety of writing tasks, and across many different writer populations, as well as to study how well findings resulting from timed-writing tasks can be generalized to writing tasks with no time restriction.

The association between cluster assignment and demographic background is worth further investigation. The study of writing process ought to integrate with the social and linguistic context [21]. Previous studies have reported subgroup differences in writing processes (e.g., between native and non-native speakers in [24], between male and female writers in [26], between black and white students in [7]). It is concievably helpful and valuable to give information about writing profiles to provide customized feedback to writers from different linguistic, social, and educational backgrounds.

Finally, although this is beyond the scope of the current study, it is worth mentioning that keystroke-enabled process visualization such as those illustrated in Figures 2 and 3, in and of itself, may have instructional value, by making it easier for students to understand and self-reflect their writing processes. For instance, teachers may select replays and graphs to demonstrate a specific writing subprocess or a writing strategy, to help the class understand how to implement a more effective writing process. Teachers might also be able to use such graphs during their one-on-one con-

ference with their students to help them better understand their writing strengths and weaknesses (such as lack of editing, low-level engagement, or lack of sufficient attention to idea generation) that are revealed by the keystroke log. Future research is encouraged to gather teacher and students feedback on the assistive value of keystroke logs in their teaching and learning experience.

5. REFERENCES

[1] P. C. Austin. An introduction of propensity score methods for reducing the effects of confounding in observational studies. *Multivariate Behavioral Research*, 46(3):399–424, 2011.

[2] V. M. Baaijen, D. Galbraith, and K. de Glopper. Keystroke analysis: Reflections on procedures and measures. *Written Communication*, 29(3):246–277, 2012.

[3] I. Breetvelt, H. van den Bergh, and G. Rijlaarsdam. Relations between writing processes and text quality: When and how? *Cognition and Instruction*, 12(2):103–123, 1994.

[4] E. F. Burke. An experimental study of the educational use of the typewriter in second grade. *Master's Thesis*, Loyola University, 1939.

[5] P. Deane and M. Zhang. Automated writing process analysis. In D. Yan, A. Rupp, and P. Foltz, editors, *Handbook of automated scoring: Theory into practice*, pages 347–364. Chapman and Hall, 2020.

[6] D. Galbraith and V. M. Baaijen. Aligning keystrokes with cognitive processes in writing. In E. Lindgren and K. P. H. Sullivan, editors, *Observing writing*, pages 306–325. Brill Publishing, 2019.

[7] H. Guo, M. Zhang, P. Deane, and R. Bennett. Writing processes differences in subgroups reflected in keystroke logs. *Journal of Educational and Behavioral Statistics*, 44(5):571–695, 2019.

[8] H. Guo, M. Zhang, P. Deane, and R. Bennett. Effects of scenario-based assessment on students' writing processes. *Journal of Educational Data Mining*, 12(1):19–45, 2020.

[9] J. R. Hayes. Modeling and remodeling of writing. *Written Communication*, 29:369–388, 2012.

[10] M. Leijten and L. V. Waes. Keystroke logging in writing research: Using inputlog to analyze and visualize writing processes. *Written Communication*, 30:358–392, 2013.

[11] D. McCutchen. A capacity theory of writing: Working memory in composition. *Educational Psychology Review*, 8:299–325, 1996.

[12] D. McPherson. A study of typing speed and accuracy development using computer-based and typewriter-based instruction in a public high school. *OTS Master's Level Project & Papers*, 352, 1995.

[13] G. W. Millgan and M. C. Cooper. A study of standardization of variables in cluster analysis. *Journal of Classification*, 5:181–204, 1988.

[14] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 502:159–179, 1985.

[15] T. Nguyen, G. S. Collins, J. Spence, J. Daures, D. P. J., P. Landais, and Y. L. Manach. Doubling-adjustment in propensity score matching analysis: Choosing a threshold for considering residual imbalance. *BMC Medical Research Methodology*, 17, 2017.

[16] T. Quinlan, M. Loncke, L. M, and L. V. Waes. Coordinating the cognitive processes of writing: The role of the monitor. *Written Communication*, 29(3):345–368, 2012.

[17] S. Sinharay, M. Zhang, and P. Deane. Prediction of essay scores from writing process and product features using data mining methods. *Applied Measurement in Education*, 32(2):116–137, 2019.

[18] C. K. Stallard. An analysis of the writing behavior of good student writers. *Research in the Teaching of English*, 8:206–218, 1974.

[19] K. P. Sullivan and E. Lindgren. *Computer keystroek logging and writing: Methods and applications.* Elsevier, New York, 2006.

[20] G. J. Szekely and M. L. Rizzo. Hierarchical clustering via joint between-within distances: Extending ward's minimum variance method. *Journal of Classification*, 22(2):151–183, 2005.

[21] L. V. Waes and P. J. Schellens. Writing profiles: The effect of the writing mode on pausing and revision patterns of experienced writers. *Journal of Pragmatics*, 35:829–853, 2003.

[22] S. Wallot and J. Grabowski. Typewriting dynamics: What dintinguishes simple from complex writing tasks. *Ecological Psychology*, 25:1–14, 2013.

[23] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[24] C. Xu and Y. Ding. An exploratory study of pauses in computer-assisted efl writing. *Language Learning & Technology*, 18:80–96, 2014.

[25] M. Zhang, R. Bennett, P. Deane, and P. van Rijn. Are there gender difference in how students write their essays? an analysis of writing processes. *Educational Measurement: Issues and Practice*, 39(2):14–26, 2019.

[26] M. Zhang, R. E. Bennett, P. Deane, and P. van Rijn. Are there gender differences in how students writer their essays? an analysis of writing processes. *Educational Measurement: Issues and Practice*, 38(2):14–26, 2019.

[27] M. Zhang, J. Hao, C. Li, and P. Deane. Classification of writing patterns using keystroke logs. In L. A. van der Ark, D. M. Bolt, W.-C. Wang, J. A. Douglas, and M. Wiberg, editors, *Quantitative psychology research: The 80th Annual Meeting of the Psychometric Society, Beijing, 2015*, pages 299–314. Springer, 2016.

Embedding navigation patterns for student performance prediction

Ekaterina Loginova
Ghent University
ekaterina.loginova@ugent.be

Dries F. Benoit
Ghent University
dries.benoit@ugent.be

ABSTRACT

Predicting academic performance using trace data from learning management systems is a primary research topic in educational data mining. An important application is the identification of students at risk of failing the course or dropping out. However, most approaches utilise past grades, which are not always available and capture little of the student's learning strategy. The end-to-end models we implement predict whether a student will pass a course using only navigational patterns in a multimedia system, with the advantage of not requiring past grades. We experiment on a dataset containing coarse-grained action logs of more than 100,000 students participating in hundreds of short course. We propose two approaches to improve the performance: a novel encoding scheme for trace data, which reflects the course structure while remaining flexible enough to accommodate previously unseen courses, and unsupervised embeddings obtained with an autoencoder. To provide insight into model behaviour, we incorporate an attention mechanism. Clustering the vector representations of student behaviour produced by the proposed methods shows that distinct learning strategies specific to low- and high- achievers are extracted.

Keywords

learning strategies, academic performance prediction, navigational patterns, mooc, lstm, autoencoder, learning management systems

1. INTRODUCTION

A large amount of trace data about learner behaviour has recently become available from online learning environments [30]. Hence, it is now possible to improve the delivery, assessment and intervention quality using data mining techniques, giving rise to technology-enhanced learning. Educators are especially interested in receiving early alerts when students are at risk of failing the course or dropping out. With these alerts, timely intervention can be organised. To estimate this risk, machine learning classification models are built to predict student performance based on their interaction with the course content. There are several ways to define student performance: it can be a binary or a multi-level grade on either the next exercise or the entire course. This study focuses on a binary grade for the given course (fail or pass), a common scenario on student performance prediction [15, 28, 27].

Traditionally, the primary source of information for this task is past academic performance records [31]. They might include the grades for the previously taken courses or intermediate test scores. However, apart from those, online learning platforms also provide information about other types of interaction between students and the content. Depending on the technical implementation and medium, it can be fine-grained click-stream video data, the text of discussion forum messages, or more coarse-grained information such as whether a person liked a video or performed a search query. Our models operate on such coarse-grained sequential records of interacting with different content types on the online learning platform and does not rely on previous grades. The motivation for it is three-fold. First, the past scores might not be available, or it can be time- and effort-consuming to provide them. For example, grading an essay usually requires a specialist, which is not scalable for MOOCs. Second, by focusing on interaction with the content, such as video views, we obtain the representation of student behaviour that is more likely to capture their learning strategies. In other words, we can discover whether they prefer a specific medium or actively interact with other students in online discussions. Third, if we work with navigation patterns, the resulting model has the potential to inform a recommendation system that would nudge a struggling student in the right direction. For instance, when students explore the platform, we can automatically recommend the next learning item to interact with (e.g., a video or a reading material they might find useful or interesting).

This study applies recurrent neural networks for academic performance prediction in short online courses. The approach we describe works without manual feature engineering or information about previous scores. In contrast to most works in the area of course grade prediction, our models operate instead on raw sequences of multiple-type interactions with the content (video view/like, discussion message, search request, exercise attempt). We propose several encoding schemes for action logs and demonstrate that they can increase classification performance. Among them, we intro-

Ekaterina Loginova and Dries Benoit "Embedding navigation patterns for student performance prediction". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 391-399. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

duce a flexible scheme that reflects relative progress within the course while being independent of its length. We also use autoencoders to extract student representation in an unsupervised way. Vector representations of student behaviour produced by different methods are clustered to compare how well they indicate (un)successful learning strategies. In order to illuminate the inner workings of the model, we provide cluster visualisations and experiment with an attention mechanism.

Thus, our research questions are the following. **RQ1**: Can we capture the information about course structure in the action sequences in a flexible way (that can be extended dynamically when new exercises or longer courses are added)? **RQ2**: Can we improve predictive performance by pretraining unsupervised embeddings (using autoencoder models)? **RQ3**: Can we improve predictive performance further by adding attention to an autoencoder?

2. RELATED WORK

While working with educational trace data, there are various ways to extract features from the raw sequences. The easiest and perhaps the most popular approach is to count the number of actions – despite its simplicity, it is a solid baseline, used even in the most recent studies [27], [28], [21].

Unfortunately, by aggregating the data in such a way, we ignore the time delay between actions and the information about their order. A common way to engineer additional, more time-sensitive features from trace data is to represent student behaviour in chunks, called studying sessions. For instance, we can define the sequence of actions as a session if less than 15 minutes passed between actions. The threshold is usually determined heuristically [20]. However, the need to choose this threshold manually is a clear drawback. Moreover, additional manual feature engineering is often required to aggregate features over sessions (e.g., experts have to define what qualifies as session intensity).

Furthermore, even though the described measures are sound and easy to use, their high volume does not necessarily contribute to the high quality of learning. Neither does it allow actionable insight beyond relatively trivial advice to spend more time in the system. As justly noted by the learning analytics community, it is the specific learning strategies adopted by individual students that are important [12].

Therefore, various approaches based on deep learning were proposed to overcome these limitations. The sequential nature of the data lends itself well to the use of recurrent neural networks (RNNs) [37]. The pioneering Deep Knowledge Tracing (DKT) model applied RNNs and its variations to the history of students' answers to predict whether they will answer the next exercise correctly [26]. One of the benefits of such neural architectures is that they can complement the educational theories proposed by human experts with insights obtained in the bottom-up, data-driven way [23]. Another benefit is that end-to-end models adapt to new domains easier and are more cost-efficient.

The success of DKT led to a new strand of research. Aiming to get rid of its simplifying assumptions, such as disregard to skill interaction or exercise text, researchers developed more advanced neural models for knowledge tracing [8].

In contrast to many grade prediction approaches such as [18], our approach does not require knowing previous academic performance or past grades. Instead, we investigate whether a binary course grade can be predicted from the trace data alone, with no intermediate exercise scores. Thus, an advantage of our approach is that we can detect low- and high-achievers without assessing the correctness of students' answers. The benefit is especially important for courses that include open-answer questions since those usually require costly human experts to be graded reliably (for example, essays in humanities subjects).

Our models work on raw sequences actions without aggregation of count or session variables. Recent studies on using RNNs on click-stream data [22], [17], [7], [16] and [18] are conceptually close to our approach in terms of using RNNs to work with sequences of actions. However, all of them but [18] operate only on video interaction and exercise answer features, whereas we also include search queries and discussion messages. [18] does not explore the autoencoders or attention and does not investigate the extracted student representation. Moreover, in most of them, aggregation still happens: [17] uses cumulative counts, [16] – weekly snapshots and in [22] interaction features are binary per item, while we feed the raw sequences as an input to predictive models. Besides, their task is to predict the next exercise response while we predict the overall course success.

Concerning the use of deep learning for unsupervised feature extraction, only a few recent publications have explored it [7]. For example, autoencoders, a popular approach in natural language processing [35], have only recently entered the educational data mining field [36]. Motivated by this, we extend recent end-to-end approaches to feature engineering on trace data by using autoencoders (including an attentional one) to embed student behaviour.

Regarding the character of the trace data used, we operate on short courses which contain multimedia content (such as videos and discussion messages; details are provided in section 3). The dataset also addresses the variability across a range of subjects [24]. Moreover, it allows us to showcase the methods in a real-life scenario, as the data is collected from a commercial educational platform. It should be noted that only coarse-grained trace data is available, i.e. there are no details of interaction with videos, such as replays. To enrich the data representation without changes to the original platform, we introduce several encoding schemes that capture the relative progress within the course.

Despite their promising results, the state-of-the-art deep learning methods are black-box models, which renders the interpretation of the prediction making process and incorporation of domain knowledge far from straightforward [10]. Such lack of interpretability can seriously hinder the adoption of otherwise efficient models in decision-critical domains such as education, as stakeholders cannot control or assess the fairness of the process. In order to illuminate how the model makes a decision, we can cluster the produced student behaviour representations or investigate attention heatmaps. In spite of the apparent success of attention mechanisms in

Table 1: An example data entry for student behaviour in a course.

user_id	77
course_id	60
avg_session_duration	584
num_sessions	5
avg_session_intensity	2.6
session_frequency	0.24
actions	[('video_view' '2015-11-13 15:33'), ...]
views	4
searches	0
messages	0
video likes	3

natural language processing [2], few researchers have utilised them for academic performance prediction so far [25].

While our primary focus is on predicting academic performance, it is also necessary to provide an insight into the learning strategies of students [12]. While this concept is reminiscent of learning styles [9], it avoids their heavily criticised assumptions [29] by performing analysis bottom-up based on raw data, instead of fitting the students in a rigid framework. One way to investigate different learning strategies is to cluster students based on their trace data. The resulting clusters implicitly classify students according to how they receive and process information or whether they are high- versus low-achievers. The obtained cluster assignment allows us to improve personalisation mechanisms since we can now use the student's preferred mode of interaction (for example, adjust the proportion of videos versus readings based on how much of a visual learner the student is). Previous research in this field relied on Hidden Markov Models, pattern mining, or Levenshtein distance between sequences [6, 13]; on numerical features, Partitioning Around Medoids [11] or k-Means are used. This study clusters the embeddings produced by predictive models and autoencoders. We expect students with similar learning strategies to appear in the same clusters. By aligning these clusters with academic performance scores, we could distinguish strategies typical for low- and high- achievers.

3. DATA & FEATURES

3.1 Data

The dataset for this study consists of student trace data extracted from an online educational platform for secondary school students (12-18 years old), with a focus on mathematics and Dutch language courses. The dataset contains interaction logs for 44 333 students and 467 short courses (177 873 students-course tuples) from 20 September 2012 to 8 August 2020. Each course includes several lessons with associated videos, discussion threads and exercises (mostly multiple choice).

The target variable is the score that the student obtains for the course. We converted the original score (from 0 to 100) into a binary variable ($I(score > 50)$), as we are interested in patterns corresponding to general success or failure in the course. Thus, for a given student-course tuple, we need to predict 0 if the student is likely to fail the course and 1 otherwise.

For this binary classification case, there is a noticeable class imbalance: there are 124 248 (70%) instances in class 1 and 53 625 (30%) in class 0. It is also important to note that the courses on the platform are rather short compared to most datasets in the field: the median number of actions per student-course tuple is 8, and the median duration of a course is approximately 14 minutes. In comparison, [7] use information about 44 920 students participating in a 4-month course focused on a single subject, with 20 interaction features.

3.2 Features

We distinguish three ways to engineer features (see Table 1 for the example data entry). Count and session features are traditional predictors. Sequences of actions are also often aggregated per timestep (e.g. a chapter in the course) instead of being used as-is. In contrast, we use the sequences in their raw, original format as input for neural models. Such a general data format can be applied in any online course with minimal technical requirements for tracking student behaviour, which benefits smaller learning platforms. More precisely, the features we use are as follows:

1. counts of actions X_c. For example, the student with id 77 watched four videos in a course with id 60. These features include the number of: video views, video likes, messages posted in forums, search queries, questions attempted (without making a distinction between correct or incorrect answer).

2. manually engineered session features X_s. We experimented with multiple threshold timeout values (as we did not have access to login and logout timestamps for students), settling on 15 minutes. We then aggregated statistics about individual studying sessions, resulting in 4 features: average session duration in seconds, number of sessions, session frequency (the ratio of number of sessions to the course duration in hours) and session intensity (average number of actions per session). This is a typical set of features engineered in similar studies [32], which we use as a benchmark.

3. raw action sequences X_a. There are five possible actions: a video view, a video like, a discussion forum message post, a search request for a term, and an attempt to answer an exercise (without the indicator of whether the answer was correct).

Due to the short duration of courses, encoding with just five types of actions without any additional information leads to low variability in data: out of 177 873 sequences, only 10 058 are unique. As a consequence, the same behaviour pattern could correspond to both passing and failing the course. To overcome this issue, we consider several ways to encode additional information in elements of the sequence. We list them by their generalisability, from least to most.

Concatenating global content item id and the corresponding actions. For instance, for a video with id 12, we would encode the action as "video_view_12" (we give examples for videos, but the idea transfers directly to other content items as well). We can also encode items associated with the video:

answering the exercise about this video would be encoded as "exercise_answer_12". The downside is that both these approaches do not generalise to new courses: we have to assume that the number of either courses or content items is fixed. Otherwise, a model needs to be retrained. It is a common problem also found in such popular approaches as DKT [33]. Besides, this approach quickly leads to an inflated vocabulary, making it computationally inefficient if one would use the bag-of-words approach (hence it is not featured in Table 2 for machine learning models, only for recurrent networks).

Encoding actions using local content item ids, preserving their relative order in the course. In other words, if the video with id 12 is the first video in a given course, then we encode the action as "video_view_1". This scheme has the potential for interesting insights into student behaviour. For example, we can see whether consequently watching the videos contributes a lot to better performance. Besides, we can gauge the engagement and background knowledge by checking whether the student immediately watched later videos. Regarding the disadvantages, the applicability to new courses is still limited: the number of videos should be the same or less than in the courses we have seen before. Using the out-of-vocabulary token or setting the maximum possible number of videos high is the simple workaround. However, a more flexible solution might be required – for which we suggest the progress percentage encoding.

Encoding the rounded percentage of the total number of videos in the course. For instance, if there are five videos in total, then the view of the second one will be represented as "video_view_40" (40%). Even though we can no longer focus on the exact content item as if the case with the other encoding schemes, we can still potentially recommend the section of the course to revise. This is the most flexible way, as it scales to new courses (of arbitrary length, with previously unseen exercises) as well.

We provide experimental results using the four proposed encoding schemes (raw X_a, global content id X_{a-gid}, local content id X_{a-lid} and progress percentage id X_{a-pid}, respectively) in section 5. To each of the above schemes, we can also add the difference in seconds with the previous action if the timestamps are available.

4. MODELS

We trained three types of classification models on numerical count features X_c and session features X_s: Logistic Regression, Decision Tree and Random Forest. For sequence data X_a, we used two popular variations of RNN – Long-Short Term Memory (LSTM) [14] and Gated Recurrent Unit (GRU) [4]. We have experimented with convolutional neural networks, but their performance was lower than that of recurrent ones. As such, for the sake of brevity, we do not focus on them in this study.

On a general level, neural classification models embed action sequences and pass them through recurrent layers to the final feed-forward layer(s) with sigmoid activation. It produces a probability of success which is then converted into a classification prediction score. A recurrent neural network takes a sequence of vectors $\{x_t\}_{t=1}^{T}$ (T is the number of timesteps) as an input and maps them to an output sequence $\{y_t\}_{t=1}^{T}$ by calculating hidden states $\{h_t\}_{t=1}^{T}$ which encode past information that is relevant for future predictions. LSTM uses a more elaborate structure in each of the repeating cells, allowing it to learn long-term dependencies. It includes so-called forget, input and output gates, which control what information to retain and pass to the next step. GRU simplifies the cell by combining the forget and input gates into a single update gate and merges the cell state and hidden state.

An encoder-decoder framework uses an RNN to read a sequence of vectors $\{x_t\}_{t=1}^{T}$ into another, context vector c in an auto-regressive fashion. If we set the desired output sequence equal to the input one – so that the goal becomes the reconstruction of the original data – we obtain an autoencoder. Then, the context vector, if its dimensionality is chosen to be lower than that of the input, will contain a denoised representation of the data that can be used in other models. This way, an autoencoder allows us to learn efficient data representation in an unsupervised manner.

Bahdanau attention mechanism allows the network to focus on certain parts of the input [2]. It is achieved by computing the context vector as a weighted sum of vectors produced by the encoder. The weights are learnt by a feed-forward neural network jointly with the rest of the model. Transformer model is a recent competitive alternative to the encoder-decoder framework [34] which foregoes the recurrent cells in favour of stacked self-attention and feed-forward layers.

5. EXPERIMENTS

We apply machine learning models on count and session features and compare them with deep learning ones on sequences of actions (with different encoding schemes used, as outlined above). Besides, we embed action sequences using an LSTM autoencoder and use its output X_{auto} as input for the classification models.

Neural network models were implemented using Keras [5] with Tensorflow backend [1] and machine learning ones with sklearn [3]. The parameters were optimised using the grid-search with stratified 10-fold cross-validation (5-fold for neural models). For recurrent neural networks, we checked the following parameter values: 32/64/128 recurrent units, 32/64/128 hidden units in feed-forward layers, 32/64/128 embedding dimensions. The maximum sequence length was set to 50. The models were trained with binary cross-entropy loss, using the Adam optimiser [19], early stopping and learning rate reduction on a plateau.

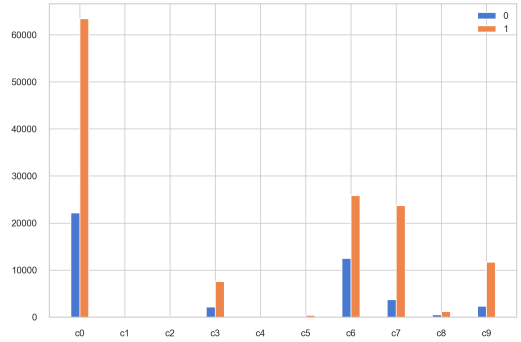
5.1 Classification

For the binary classification task, the cross-validation ROC AUC scores are presented in Table 2. Inspecting the table, we can conclude that end-to-end models perform at least as well as the ones using manually engineered features, even when the length and variability of actions sequences are limited.

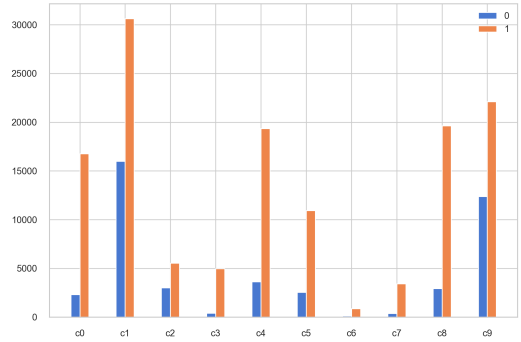
Concerning **RQ1**, the contribution of different encoding schemes, we can see that global content id encoding performs the best. However, as mentioned above, it does not scale to new content items. However, using the percentage encoding scheme

Table 2: Cross-validation ROC AUC scores for classification models. Input features: count features X_c, session features X_s, action sequences X_a with encoding scheme variations (no id X_a, global content id X_{a-gid}, local content id X_{a-lid}, progress percentage id X_{a-pid}), actions embeddings by LSTM autoencoder X_{auto}, actions embeddings by an attentive (Bahdanau) LSTM autoencoder X_{auto_B}.

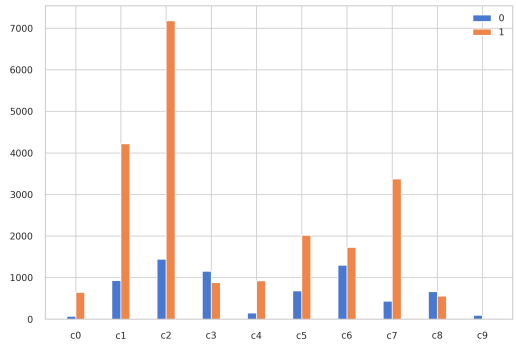
	X_c	X_s	X_{a-lid}	X_{a-pid}	X_{auto}	X_{auto_B}
Logistic Regression	0.62	0.62	0.69	0.67	0.77	0.76
Random Forest	0.73	0.81	0.81	0.81	0.82	0.81
Decision Tree	0.73	0.80	0.79	0.80	0.80	0.80
	X_a	X_{a-gid}	X_{a-lid}	X_{a-pid}	X_{auto}	X_{auto_B}
LSTM	0.73	0.88	0.82	0.81	0.82	0.83
GRU	0.73	0.87	0.81	0.81	0.83	0.82



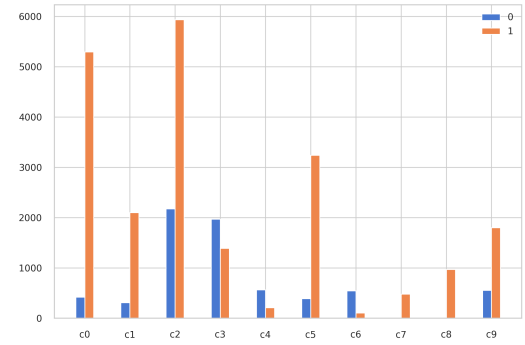
(a) X_c (count features)



(b) X_s (session features)



(c) X_a (LSTM embeddings on action data)



(d) X_{a-pid} (LSTM embeddings on action data with progress percentage id)

Figure 1: Distribution of target variable (pass/fail score) over the K-Means clusters based on different data representations (with Euclidean distance; K of 3, 5, 10 and 20 were tried). LSTM embeddings distinguish better between high- and low-achievers, producing clusters that clearly correspond to one class more than the other, while for traditional count (a) and session (b) data, most of the clusters contain a mix of both classes. For LSTM embeddings on action data (c), clusters appear that contain more failing students than passing – and are thus more useful for early warning systems. The effect is even more pronounced if we use progress percentage encoding (d).

also gives a steady increase in ROC AUC score: from 0.73 to 0.81 for Random Forest, where we use count features, and similarly for recurrent networks, where we use sequences. Using unsupervised embeddings obtained with an autoencoder (the focus of **RQ2**), we gain an improvement as well: from 0.73 to 0.82 for both machine and deep learning models.

Contrary to our expectations for **RQ3**, adding attention did not significantly improve the results. We experimented with including an attention mechanism directly into the classification models, both Bahdanau and Transformer. It can be done, for example, by training a simple Transformer, then using the average of its encoder's hidden states as an input to a classification model on top. Unfortunately, those modifications did not influence performance in our experiments, in contrast to [25]. We hypothesise that the features might be too coarse-grained and the sequences too short to take full advantage of the technique; there are also no skill labels that would provide the hierarchical structure that attention mechanisms can reflect. However, using attention allows us to produce visualisations that a first step to understanding the decision-making process of the neural network. Thus, we provide the attention scores from a Bahdanau-attention classification model below for illustration purposes.

To gain more actionable insight from our models, we also investigated the predictive performance on partial action sequences. Being able to predict early that a student will fail the course would allow sending a timely alert to the educator, signalling the need for intervention. Hence, we explored how the performance of the models changes when only the first N actions are available. For these experiments, to ensure that the model does not have full information, we only used courses with more than five questions and more than eight actions. As depicted in Figure 2, sequential data lead to higher scores than count features and proposed encoding schemes outperform raw action sequences.

5.2 Clustering & Visualisation

We clustered the student-course tuples based on different data representations using k-Means and plotted the distribution of the target class over these clusters (see Figure 1d) to investigate how the models distinguish between learning strategies of low- and high-achievers. For traditional count and session data, most of the clusters are not easily interpretable, as they contain a mix of both classes, roughly following the target label distribution. For instance, when using count features, almost all student-course pairs are in just three clusters, so there is little distinction between learning strategies. It should be noted that even though some clusters appear empty, in fact they still contain a very small number of students.

When we increase the representation's complexity, the extracted groups are more distinct. The distribution of high- and low-achievers in them shifts so that clusters with the prevalence of a single class emerge. The improvement is even more noticeable with progress percentage encoding: more clusters are extracted where one class is prevalent. For LSTM embeddings on action data, clusters appear that contain more failing students than passing, signalling that this is an unsuccessful strategy (clusters 4 and 6 on 1). This information is vital for early warning systems. The effect is even more pronounced if we use the progress percentage encoding, which encourages the application of these schemes for distinguishing between successful and unsuccessful learning strategies.

Another way to shed light on the prediction process of a neural network is to visualise attention heatmaps, where higher scores correspond to actions in the sequence that are more important for the classification decision (Figure 3).

Limitations & Future work. A wide range of topics covered in these courses might influence the performance, as different subjects are likely to demand different behaviour patterns to successfully pass the course (e.g., humanities versus technical subjects). We plan to investigate them separately and include information about exercise content. Finally, we plan to train a recommendation system informed by the extracted learning strategies to aid navigation.

6. CONCLUSION

We show that it is possible to predict whether students will pass the course using only their navigation pattern sequence in the online learning platform, without information about past grades. The findings of our study suggest that features extracted with deep learning are efficient even if the courses are extremely short, cover multiple different subjects, and only a limited number of interaction types is available.

We propose a flexible way to increase the classification performance with minimal preprocessing of the raw sequences of actions extracted from the learning platform required. A novel and relatively simple percentage progress encoding scheme is introduced which captures the course structure while scaling well to the new data. It results in an improvement of almost 10% in ROC AUC score. We also demonstrate a positive effect of using pre-trained unsupervised embeddings obtained with autoencoders (up to 15% improvement when using in machine learning models, compared to traditional features). We cluster resulting embeddings to show that using action sequences has more potential for distinguishing between strategies specific to high and low achievers than simple count or session features. It is possible to visualise attention heatmaps and see the contribution of individual actions to the classification decision to interpret retrieved strategies.

Our research supports decision-makers, as it allows detecting (un)successful students from their navigation patterns alone, without having to grade intermediate exercises. The action sequences corresponding to high achievers can be used to inform learning design patterns and recommendation systems in a more meaningful way than the standard count features. As the proposed models are shown to outperform several baselines on extremely short, incomplete action sequences, they allow us to intervene early if a student begins to follow a trajectory associated with a lower chance of success.

7. REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser,

M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. 2015.

[3] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[4] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[5] F. Chollet et al. Keras. https://keras.io, 2015.

[6] M. C. Desmarais and F. Lemieux. Clustering and visualizing study state sequences. In S. K. D'Mello, R. A. Calvo, and A. Olney, editors, *Proceedings of the 6th International Conference on Educational Data Mining, Memphis, Tennessee, USA, July 6-9, 2013*, pages 224–227. International Educational Data Mining Society, 2013.

[7] M. Ding, K. Yang, D. Yeung, and T. Pong. Effective feature learning with unsupervised learning for improving the predictive models in massive open online courses. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge, LAK 2019, Tempe, AZ, USA, March 4-8, 2019*, pages 135–144. ACM, 2019.

[8] X. Ding and E. C. Larson. Why deep knowledge tracing has less depth than anticipated. 2019.

[9] J. Feldman, A. Monteserin, and A. Amandi. Automatic detection of learning styles: state of the art. *Artif. Intell. Rev.*, 44(2):157–186, 2015.

[10] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 3449–3457. IEEE Computer Society, 2017.

[11] D. Furr. Visualization and clustering of learner pathways in an interactive online learning environment. In M. C. Desmarais, C. F. Lynch, A. Merceron, and R. Nkambou, editors, *Proceedings of the 12th International Conference on Educational Data Mining, EDM 2019, Montréal, Canada, July 2-5, 2019*. International Educational Data Mining Society (IEDMS), 2019.

[12] D. Gašević, S. Dawson, and G. Siemens. Let's not forget: Learning analytics are about learning. *TechTrends*, 59(1):64–71, 2015.

[13] N. Gitinabard, T. Barnes, S. Heckman, and C. F. Lynch. What will you do next? A sequence analysis on the student transitions between online platforms in blended courses. 2019.

[14] S. Hochreiter and J. Schmidhuber. LSTM can solve hard long time lag problems. In M. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, pages 473–479. MIT Press, 1996.

[15] B. Jeon, E. Shafran, L. Breitfeller, J. Levin, and C. P. Rosé. Time-series insights into the process of passing or failing online university courses using neural-induced interpretable student states. 2019.

[16] H. Karimi, T. Derr, J. Huang, and J. Tang. Online academic course performance prediction using relational graph convolutional neural network. In A. N. Rafferty, J. Whitehill, C. Romero, and V. Cavalli-Sforza, editors, *Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020, Fully virtual conference, July 10-13, 2020*. International Educational Data Mining Society, 2020.

[17] T. Käser and D. L. Schwartz. Exploring neural network models for the classification of students in highly interactive environments. 2019.

[18] B. Kim, E. Vizitei, and V. Ganapathi. Gritnet: Student performance prediction with deep learning. 2018.

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2015.

[20] V. Kovanovic, D. Gasevic, S. Dawson, S. Joksimovic, R. S. Baker, and M. Hatala. Penetrating the black box of time-on-task estimation. In J. Baron, G. Lynch, N. Maziarz, P. Blikstein, A. Merceron, and G. Siemens, editors, *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge, LAK '15, Poughkeepsie, NY, USA, March 16-20, 2015*, pages 184–193. ACM, 2015.

[21] B. Mbouzao, M. C. Desmarais, and I. Shrier. A methodology for student video interaction patterns analysis and classification. In M. C. Desmarais, C. F. Lynch, A. Merceron, and R. Nkambou, editors, *Proceedings of the 12th International Conference on Educational Data Mining, EDM 2019, Montréal, Canada, July 2-5, 2019*. International Educational Data Mining Society (IEDMS), 2019.

[22] K. Mongkhonvanit, K. Kanopka, and D. Lang. Deep knowledge tracing and engagement with moocs. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge, LAK 2019, Tempe, AZ, USA, March 4-8, 2019*, pages 340–342. ACM, 2019.

[23] S. Montero, A. Arora, S. Kelly, B. Milne, and M. Mozer. Does deep knowledge tracing model interactions among skills? 2018.

[24] B. Motz, J. Quick, N. L. Schroeder, J. Zook, and M. Gunkel. The validity and utility of activity logs as a measure of student engagement. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge, LAK 2019, Tempe, AZ, USA, March 4-8, 2019*, pages 300–309. ACM, 2019.

[25] S. Pandey and G. Karypis. A self attentive model for knowledge tracing. 2019.

[26] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami,

L. J. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 505–513, 2015.

[27] A. Pigeau, O. Aubert, and Y. Prié. Success prediction in moocs: A case study. 2019.

[28] A. Polyzou and G. Karypis. Feature extraction for classifying students based on their academic performance. In K. E. Boyer and M. Yudelson, editors, *Proceedings of the 11th International Conference on Educational Data Mining, EDM 2018, Buffalo, NY, USA, July 15-18, 2018*. International Educational Data Mining Society (IEDMS), 2018.

[29] C. Riener and D. Willingham. The myth of learning styles. *Change: The magazine of higher learning*, 42(5):32–35, 2010.

[30] C. Romero and S. Ventura. Data mining in education. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 3(1):12–27, 2013.

[31] C. Romero and S. Ventura. Educational data mining and learning analytics: An updated survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 10(3), 2020.

[32] A. Sheshadri, N. Gitinabard, C. F. Lynch, T. Barnes, and S. Heckman. Predicting student performance based on online study habits: A study of blended courses. *CoRR*, abs/1904.07331, 2019.

[33] S. Sonkar, A. S. Lan, A. E. Waters, P. Grimaldi, and R. G. Baraniuk. qdkt: Question-centric deep knowledge tracing. 2020.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[35] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 352–357. The Association for Computer Linguistics, 2015.

[36] Y. Wang, N. Law, E. Hemberg, and U. O'Reilly. Using detailed access trajectories for learning behavior analysis. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge, LAK 2019, Tempe, AZ, USA, March 4-8, 2019*, pages 290–299. ACM, 2019.

[37] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280, 1989.

APPENDIX

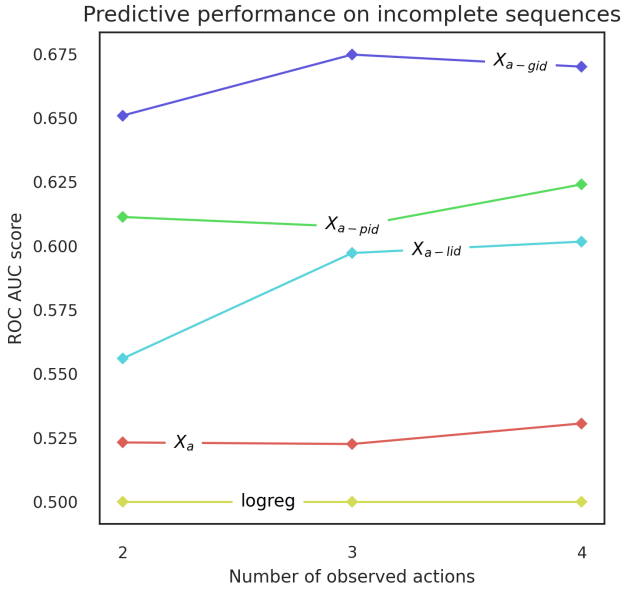


Figure 2: Comparison of validation ROC AUC scores of an LSTM on encoding schemes in the incomplete sequence scenario. We can see that percentage and local content id encoding schemes perform better than raw actions. As such, we would be able to detect whether a student is likely to fail from the first two actions already.

Figure 3: An attention heatmap of the RNN model with Bahdanau attention mechanism on action data (multiple sequences view). Higher scores (brighter colours) correspond to actions in the sequence that are more important for classifying the student as passing or not. If we use an encoding scheme which includes content id (such as the global content id here), we see which content items contribute more to the classification decision: for example, in the bottom row, the viewing of the video with id 264231 is more important for the network than the others.

Assessing Attendance by Peer Information

Pan Deng[1], Jianjun Zhou[1*], Jing Lyu[2], Zitong Zhao[2]
[1]Shenzhen Research Institute of Big Data, Shenzhen, China
[2]The Chinese University of Hong Kong, Shenzhen, China
{pandeng, jinglyu, zitongzhao}@link.cuhk.edu.cn
zhoujianjun@cuhk.edu.cn

ABSTRACT

Attendance rate is an important indicator of students' study motivation, behavior and Psychological status; However, the heterogeneous nature of student attendance rates due to the course registration difference or the online/offline difference in a blended learning environment makes it challenging to compare attendance rates. In this paper, we propose a novel method called Relative Attendance Index (RAI) to measure attendance rates, which reflects students' efforts on attending courses. While traditional attendance focuses on the record of a single person or course, relative attendance emphasizes peer attendance information of relevant individuals or courses, making the comparisons of attendance more justified. Experimental results on real-life data show that RAI can indeed better reflect student engagement.

Keywords

attendance, peer information, engagement, academic performance, comparison, clustering

1. INTRODUCTION

While studying offline is the norm for most schools, during epidemic periods all or a portion of students are forced to study online due to university closure. In such a blended learning environment, tracking the study status and the wellbeing of students is an important issue for the university. Students' attendance in classes is a measure that reflects students' enthusiasm for the course and their status in the university [29]. Many studies suggest a correlation between attendance and attainment at university [5, 26, 14]. Several studies detect attendance rates using mobile devices and include attendance as a feature to predict academic performance [27, 19, 28]. Attendance is also correlated with behavior and Psychological problems such as video game addiction [24] and depression [26]. Detecting unusual attendance rate changes can help to identify abnormal behaviors and Psychological problems in an early stage and provide in time intervention to students in need.

The successful applications of attendance data call for fair comparisons among peer attendance, especially in universities. Traditionally, the attendance rate of a course or a student is isolated and might not be compared fairly, because in many universities students are allowed to select some courses on their own so that the course registration records of two students can be different. In addition, each course can have its own attendance policy, making it harder to compare attendance rates. Courses that have mandatory attendance requirements usually have higher attendance rates than those do not, so that it is not fair to compare course attendance rates without considering attendance requirements. Similarly, students who registered for courses with mandatory attendance requirements usually have higher attendance rates than students who registered for courses with voluntary attendance requirements, so that the attendance rate does not always reflect the attainment of a student. Attendance of online and offline courses may not be compared directly as well, because the efforts to attend those courses can be significantly different. Attending online courses could be as simple as a mouse click away, while attending offline courses usually requires travelling from place to place physically.

Traditionally it is not easy to fairly compare attendance in a university, due to not just the diversity of course registration and attendance requirements but also the difficulty of collecting campus-wide attendance data. Without attendance information of peer students or courses, the attendance data of a student or a course is isolated and difficult to adjust. However, in the era of Big Data, many new technologies [12, 18, 31] have been proposed to collect attendance data for many courses simultaneously, making it possible to analyze the attendance structure of the student population, and develop new attendance calculation methods.

Careful comparisons of attendance can also provide insight into students' academic interest. If a student attends a course that has a generally low attendance rate, it indicates that the student is more willing to attend the course than their classmates are; On the other hand, if a student attends a course that has a generally high attendance rate, it indicates that the student is just doing what others are doing.

In this paper, we propose a novel method called Relative Attendance Index (RAI) to measure attendance, which reflects the efforts on attending courses and makes comparisons of attendance more justified. To our knowledge, this is the first study on fair comparisons of attendance. While traditional attendance focuses on the record of a single person or course, we define a notion for attendance contribution to course attendance and add the attendance information of relevant individuals or courses to make the comparisons of attendance fairer. We perform a campus-wise study on attendance and analyze its effects on course grades and GPA. Our experiment results show that RAI has a higher correlation with academic performance than the traditional attendance rate.

Pan Deng, Jianjun Zhou, Jing Lyu and Zitong Zhao "Assessing attendance by peer information". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 400-406. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

* The corresponding author.

The rest of this paper is organized as follows. Section 2 describes the related studies. Section 3 introduces the RAI definition. Section 4 presents the experiment results on real-life data from a university. Section 5 discusses an application of RAI on clustering student populations. Section 6 describes the limitations and future work. Section 7 lists the acknowledgments.

2. RELATED STUDIES

With the advancement of technology, many new methods have been proposed to collect campus-wide attendance data. Several studies [13, 11, 18] measured attendance via QR code systems in which QR codes are generated and then scanned by students to authenticate themselves. Wang et al. [26] deployed an APP to students' cell phones to detect attendance by GPS signals and WiFi tracing. A method independently developed in [19] and [28] used WiFi log to calculate attendance. Studies in [2, 25] proposed Bluetooth/Beacon based attendance prediction systems. Shoewu and Idowu [22] used fingerprints and Kar et al. [12] used face recognition to detect individual attendance rates.

Some studies measured offline and online attendance at the same time. Brennan et al. [4] detected physical attendance by thermal sensors and online behaviors by clickstream data. The change of online and physical attendance through time was observed. However, due to a technology limit, the method did not link physical attendance to individuals and did not study the issue of attendance comparison. Nordmann et al. [17] mixed the data of physical attendance and online recording clickstream together to form the total attendance rate instead of studying them separately. However, attendance of live lectures is still a stronger predictor than recording use on students' academic performance.

Many studies confirmed the correlation between attendance and academic performance [1, 3, 7, 16]. See [15] for a survey. [15] also reviewed factors that affect attendance. To work around the issue of fair comparisons of attendance, many studies focused on samples from the same course or samples with similar registration records (e.g., first year students) [1, 3, 10]. [3] also controlled factors such as age, gender, nationality etc. in their regression analysis. Studies in [7] and [16] divided the students into bands according to grades and used the average attendance of each band for correlation studies.

Student subtyping and clustering are widely used in analyzing learning process and predicting academic performance. Yang et al. [30] applied EM-IRL to students learning behavior data and observed significant differences between groups. Romero et al. [20] used clustering on online forum data to predict students' final performance. Resulting model turned out to be suitable and highly interpretable. Cerezo et al. [5] studied both learning process and clusters' relation with performance using LMS logs data. Resulting clusters are well-interpreted and showed satisfying correlation with final marks.

Many studies explored the reasoning for student class attendance. Friedman et al. [9] and Moore et al. [14] reported positive relationship between class attendance and students' motivation. Sloan et al. [23] further found that the level of interest has significant impact on attendance. These studies indicated that attendance, along with other features, can better show students' academic interest than traditional models.

None of the above studies has applied peer information to revise attendance measurements.

3. METHOD

The traditional attendance rate of a class or a student is defined in a straightforward way. Only the information of the class or the student is involved. We give the definition of Attendance Rate (AR) formally as in Definition 1.

Definition 1 (Attendance Rate r_c and r_s): Given class c and student s, let n_c^{reg} and n_c^{att} be the number of students registered c and the number of students attended c respectively; let n_s^{reg} and n_s^{att} be the number of classes registered by s and the number of classes attended by s. Then the Attendance Rate (AR) of class c (r_c) and of student s (r_s) are defined as below respectively.

$$r_c = \frac{n_c^{att}}{n_c^{reg}}, \tag{1}$$

$$r_s = \frac{n_s^{att}}{n_s^{reg}} \tag{2}$$

Students can have different sets of registered classes, and classes can have very different attendance requirements. When comparing the attendance rates of two students, it is necessary to analyze the set of classes attended by these two students and the attendance rates of these classes. If a student attends a class attended by almost everyone, the student makes little contribution to the attendance rate of the class; on the other hand, if a student attends a class that has a low attendance rate, the student makes a significant contribution to the attendance rate. To capture the concept, we propose the notion of attendance contribution as in Definition 2.

Definition 2 (Attendance Contribution D_{sc}): Let r_c be the attendance rate of class c, and a_{sc} be a function indicting whether student s attended class c or not, then the Attendance Contribution of student s on the attendance rate of class c is defined as

$$D_{sc} = a_{sc} - r_c, \text{ with} \tag{3}$$

$$a_{sc} = \begin{cases} 1, & \text{if } s \text{ attended } c \\ 0, & \text{if } s \text{ did not attend } c \end{cases} \tag{4}$$

Since $r_c \in [0, 1]$, Attendance Contribution is a number between -1 and 1. If s has registered c and s attended c, then the attendance rate of c cannot be zero and D_{sc} can approach 1 but never reach 1.

With Attendance Contribution, we can compare the attendance rates of two students by computing the average Attendance Contribution on registered classes. We defined the notion as Relative Attendance Index (RAI) in Definition 3.

Definition 3 (Relative Attendance Index RAI_s**):** Given student s, Let K_s be the set of classes registered by s, the Relative Attendance Index (RAI) of s is defined as

$$RAI_s = \frac{\sum_{c \in K_s} D_{sc}}{|K_s|} \tag{5}$$

RAI considers both the student's individual attendance status of a semester and the attendance status of the student's classmates. The peer information is injected into the new measure through the course attendance rate in Attendance Contribution.

LEMMA 1: $-1 < RAI_s < 1$.

Proof: The RAI_s definition only considers classes registered by s. When $a_{sc} = 0$, $r_c \in [0,1)$; When $a_{sc} = 1$, s attended c, therefore $r_c \in (0,1]$. Thus $a_{sc} - r_c \in (-1, 1)$, Therefore $RAI_s = \frac{\sum_{c \in K_s}(a_{sc} - r_c)}{|K_s|} \in (\frac{|K_s| \times -1}{|K_s|}, \frac{|K_s| \times 1}{|K_s|})$ = (-1, 1). □

RAI is a number between -1 and 1. When RAI approaches -1, a_{sc} is mostly 0 and r_c approaches 1 for most classes, indicating that

the student has skipped many well attended classes. On the other hand, when RAI approaches 1, a_{sc} is mostly 1 and r_c approaches 0 for most classes, indicating that the student has attended many poorly attended classes. Therefore, RAI shows the difference of attitudes toward classes between students and their classmates.

4. Results

4.1 Data and Setup

The anonymous attendance and grade data used in this paper were collected in 2018 and 2019 from a university[1] in China. We applied the method proposed in [19] and [28] to calculate the attendance. Note that our Relative Attendance Index can be applied to attendance data collected using other methods such as QR code [18]. The student IDs were converted into hash codes, then the attendance and grade data were connected through the hash codes. The university did not have a mandatory attendance policy; however, while most instructors followed the university policy, some instructors had their own attendance requirements. Some instructors used in-class discussions and quizzes to encourage attendance. The data came from 4838 students from Cohort 14 to 19 in 44 majors, spanning over 3 semesters, with 1489 courses grouped into 37 categories by the university. For most courses, students received letter grades from A to F. Courses with other grades such as P/F were excluded. The traditional attendance rate (AR) for a student was calculated using Formula (2) in Definition 1, and the corresponding RAI was calculated using Formula (5) in Definition 3.

4.2 Correlation with Academic Performance

Many previous studies show that attendance is correlated with academic performance. Given that the purpose of student attendance comparison is usually to assess the attainment of the students, we calculated the correlation between attendance rates and GPA to assess the fairness of attendance comparisons. A more correctly calculated attendance assessment will have a higher correlation with the GPA. The Pearson correlation between RAI and GPA is 0.48, which is significantly higher than that of AR (0.37). The p-values of the two correlation values are 3.7×10^{-225} and 2.6×10^{-129} respectively. Since they are well below the 0.05 threshold, the correlation values are generally considered significant.

We also calculated the correlation between attendance and academic performance within each course category. The result is shown in Table 1 (sorted by the RAI correlation). Some course categories were filtered out because they had small enrollments and did not generate correlation values with low enough p-values (<0.05) to be statistically significant. For 19 out of the 26 course categories, RAI has a higher correlation than AR. Only for two categories, GED and FRN, AR has a higher correlation than RAI (The descriptions of the categories are listed in Table 1). AR and RAI are tie for the five categories of FMA, GNB, ERG, CHM and CSC. We remark that language related courses such as ENG (English) have low correlations because those courses usually have in-class discussions resulting in an unofficial mandatory attendance requirement. Categories that rely on prior knowledge in high school, such as Chemistry and Physics, also have low correlations. Since most students attending this university did not study Calculus in high school and Calculus accounts for a large faction in Mathematics courses, it is reasonable to see the MAT category having a much higher correlation.

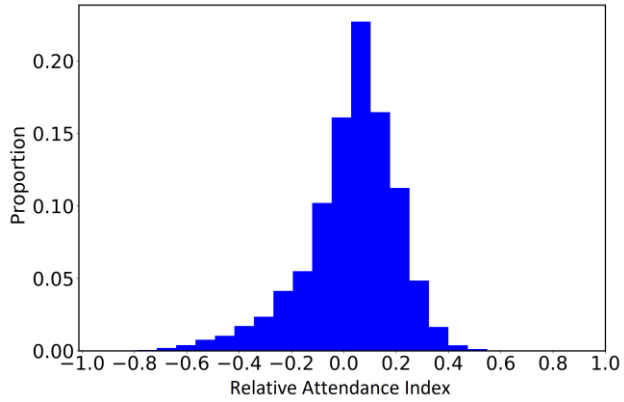
Table 1. Correlation in course categories

CAT.	Description	AR	RAI
FMA	Financial Mathematics	**0.65**	**0.65**
MSE	Material Science and Engineering	0.46	**0.52**
BIM	Bioinformatics	0.35	**0.51**
GED	General Education D	**0.48**	0.46
GEB	General Education B	0.42	**0.45**
STA	Statistics	0.32	**0.39**
MGT	Management	0.26	**0.39**
GFN	GE Foundation: In Dialogue with Nature	0.28	**0.37**
FIN	Finance	0.34	**0.36**
MAT	Mathematics	0.34	**0.36**
EIE	Electronic Information Engineering	0.34	**0.36**
GFH	GE Foundation: In Dialogue with Humanity	0.34	**0.36**
ECO	Economics	0.27	**0.36**
GNB	Genomics and Bioinformatics	**0.35**	**0.35**
GEA	General Education A	0.32	**0.35**
ACT	Accounting	0.32	**0.34**
PHY	Physics	0.22	**0.29**
HSS	Humanities and Social Science	0.27	**0.28**
ERG	General Engineering courses	**0.27**	**0.27**
GEC	General Education C	0.24	**0.27**
CHM	Chemistry	**0.25**	**0.25**
FRN	French	**0.28**	0.23
CSC	Computer Science	**0.23**	**0.23**
MKT	Marketing	0.12	**0.22**
CHI	Chinese	0.13	**0.19**
ENG	English	0.06	**0.12**

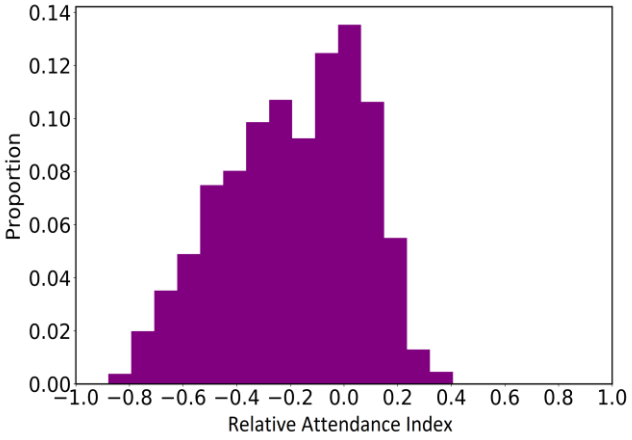
4.3 RAI Distribution

To illustrate the different distributions on RAI for high and low course grade students, we collected two sets of samples, with one set having a course grade no less than B+ and the other set no greater than C. Each sample is a triplet with a hashed student ID, a course ID, and the corresponding grade received by the student in the course. We then calculated the RAI of the student in the corresponding course. Figure 1 (a) shows the distribution of the first set. It shows that more than 50% of samples have RAI > 0 (better than normal). Figure 1 (b) shows the distribution of the low course grade set. It shows that the majority of samples have RAI < 0 (worse than normal), with some down to -0.8. For easier comparisons of both sets, the values in both subfigures have been

[1] The use of the data by our project has been approved by the university management and the committee in charge of personal information in this university.



(a) Samples with grades ≥ B+



(b) Samples with grades ≤ C

Figure 1. RAI of high and low course grade samples.

normalized as the proportion values. We can see that the first set has a more concentrated distribution than the second set. This indicates that students receiving grade C or lower have a much higher probability of having extreme attendance behaviors (skipping many courses).

5. DISCUSSION

In this section, we showcase an application of RAI on clustering the student population.

We formatted the attendance values in the 37 course categories as a vector for each student, then applied a clustering algorithm on the vectors. Since the dimensionality of 37 is too high for most clustering algorithms, we applied PCA to reduce the dimensionality. The clustering algorithm we applied was the DBSCAN clustering algorithm [8] using the Euclidean distance. DBSCAN performs density-based clustering and does not require the input of the cluster number. The parameters we tuned in this experiment are specified in Table 2. We applied silhouette score [21] to select the best set of parameters with the highest silhouette score.

Table 2. Parameters to tune for the clustering.

Parameters	Range
Number of PCA components	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Eps of DBSCAN	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
MinPoints of DBSCAN	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

We applied the same clustering procedure to AR and RAI attendance values respectively. For AR, the procedure failed to generate meaningful clustering (the result contained one big cluster only). For RAI the procedure produced 8 clusters with 616, 472, 665, 520, 130, 1799, 61, and 549 students respectively. 26 student samples were labeled as noise by DBSCAN and excluded in the follow-up study. For each cluster, we identified the top five most popular majors among the samples in the cluster to analyze students' academic interest and performance.

Figure 2 shows the profiles of the 8 clusters from the RAI clustering, labeled as cluster 1 to 8. We combined the dimensions of student numbers, distribution of majors, RAI attendance rates, top 10% academic performance ratio, and last 10% academic performance ratio to show how RAI attendance is related with academic performance and how the analysis can provide guidance on major selection. While some of the findings are interesting, we admit that not all phenomena can be fully explained due to the complexity behind attendance and attainment [15]. For all subfigures in Figure 2, the X is the major of the students. Figure 2(a) shows the number of students in each major for the 8 clusters in a row. Some of the clusters are very specific. Cluster 5 contains two majors only, TRAN (Translation) and PSY (Psychology); Cluster 7 contains the major of FE (Financial Engineering) only. Figure 2(b) shows the distribution of majors among the clusters (whether a cluster accounts for a significant portion of the students in a major), with each bar representing a fraction of the corresponding major in the university. For example, as shown in Figure 2(b), close to 70% of the students majoring in PSY are in cluster 5; close to 50% of the students majoring in CSE (Computer Science and Engineering) are in Cluster 6, with other large portions of CSE students in Cluster 2, 3 and 4. Figure 2(c) shows the RAI attendance of the clusters. Students in Cluster 6 have significantly lower RAI values than the other clusters. Figure 2(d) shows the ratio of students with a GPA in the top 10% of the major. If a bar of major *m* is higher than the 0.1 line, it means that the students from the cluster in major *m* outperform the average level of students in major *m*. Similarly, Figure 2(e) shows the portion of students with a GPA in the last 10% of the major. The higher the value, the worse the performance of the students, which is the opposite of Figure 2(d).

Figure 2 illustrates how RAI correlates with academic performance. Figure 2(c) shows that Cluster 6 has the overall lowest RAI values, with all the five majors having negative RAI values. Cluster 6 also has the worst top 10% ratio in Figure 2(d) (only one major is barely over the average cutline), and the worst last 10% ratio in Figure 2(e) (all five majors worse than the average). The TRAN major has about the same number of students in Cluster 5 and Cluster 8. The TRAN in Cluster 8 has a higher RAI value as well as a higher top 10% ratio and a much lower last 10% ratio than TRAN in Cluster 5. There are exceptions though. CSE in Cluster 3 has a negative RAI, but its

top 10% ratio is the highest in Cluster 3. However, this is consistent with our result in Table 1, which shows that CSE courses have a relatively low RAI correlation with academic performance (CSE major students usually take many CSE courses).

Another interesting finding is that in all the 7 clusters with more than one major, the major that has the highest RAI value also has the lowest last 10% ratio except for Cluster 2. In Cluster 2 it is the second highest RAI major EIE that has the lowest last 10% ratio. The highest RAI major in Cluster 2 is BIFC (Bioinformatics), a new major with a relatively small enrollment. Students facing the risk of poor academic performance may consider selecting or switching to the major with the highest RAI in the same cluster. While we admit that this is by-no-mean a correlation between RAI and students' academic interest, we remark that the interest in a subject is generally believed to be a weapon to fight against poor performance. Together with the fact that DBSCAN worked better on RAI than AR, we believe this phenomenon may suggest that RAI has a better potential than AR for exploring students' academic interest.

6. LIMITATIONS AND FUTURE WORK

In this study, we defined the Relative Attendance Index to adjust the attendance measurement, with the objective of better reflecting students' attainment and interest. While attendance is affected by many factors [15], the new information we introduced is only the attendance of the peer. Further improvements should address more factors of attendance.

When clustering on student data, the clustering algorithm DBSCAN worked better on RAI than AR data, and the clustering analysis confirmed the correlation between RAI and academic performance. We admit that we have not been able to confirm the correlation between RAI and students' academic interest, which is an interesting topic to be further explored.

The raw attendance data was collected using a WiFi based method [19, 28]. It is possible that some students closed the WiFi connection on their cell phones or even closed their cell phones all together before class, leading to a false label of absence. If a student had less than 50 WiFi connection records in a week, their data in that week were excluded from the statistics. While we admit that this could generate some noise in the attendance, we

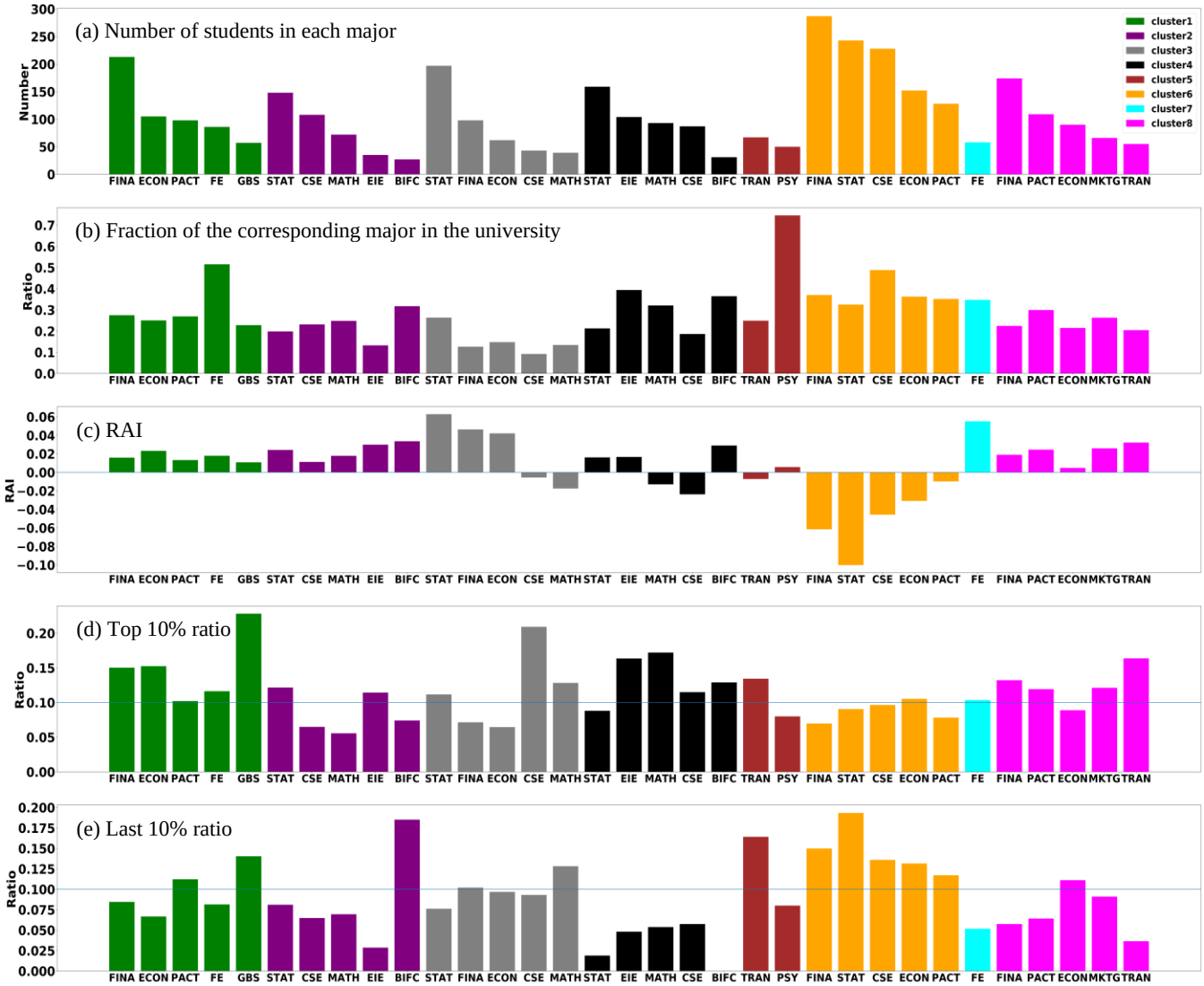


Figure 2. Student Clustering.

observed that this situation only occurred rarely. For example, in Fall term 2019 we found that only 2 out of 4838 students turned off WiFi completely, and only 3.34% of the students had some weeks of data filtered out. We conjecture two reasons for this phenomenon. First, usage of laptops and tablets is popular among the students in this university. Many students carry them to the classroom to view course materials. Laptops and tablets usually can connect WiFi only. Secondly, students use time confetti before and after class to chat with friends or read the news. It will be interesting to see how RAI performs on other attendance data such as QR code scanning and online attendance data.

7. ACKNOWLEDGMENTS

This work was supported by Shenzhen Research Institute of Big Data. We also want to thank anonymous reviewers for helpful suggestions.

8. REFERENCES

[1] Alexander, V., & Hicks, R. E. (2016). Does class attendance predict academic performance in first year psychology tutorials. International Journal of Psychological Studies, 8(1), 28-32.

[2] Avireddy, S., Veerapandian, P., Ganapati, S., Venkat, M., Ranganathan, P., & Perumal, V. (2013, March). MITSAT—An automated student attendance tracking system using Bluetooth and EyeOS. In 2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s) (pp. 547-552). IEEE.

[3] Bijsmans, P., & Schakel, A. H. (2018). The impact of attendance on first-year study success in problem-based learning. Higher Education, 76(5), 865-881.

[4] Brennan, A., Sharma, A., & Munguia, P. (2019). Diversity of Online Behaviours Associated with Physical Attendance in Lectures. Journal of Learning Analytics, 6(1), 34-53.

[5] Cerezo, R., Sánchez-Santillán, M., Paule-Ruiz, M. P., & Núñez, J. C. (2016). Students' LMS interaction patterns and their relationship with achievement: A case study in higher education. Computers & Education, 96, 42-54.

[6] Clark, G., Gill, N., Walker, M., & Whittle, R. (2011). Attendance and performance: Correlations and motives in lecture-based modules. Journal of Geography in Higher Education, 35(2), 199-215.

[7] Colby, J. (2005). Attendance and Attainment-a comparative study. Innovation in Teaching and Learning in Information and Computer Sciences, 4(2), 1-13.

[8] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd (Vol. 96, No. 34, pp. 226-231).

[9] Friedman, P., Rodriguez, F., & McComb, J. (2001). Why students do and do not attend classes: Myths and realities. College teaching, 49(4), 124-133.

[10] Fryer, L. K., Ginns, P., Howarth, M., Anderson, C., & Ozono, S. (2018). Individual differences and course attendance: why do students skip class?. Educational Psychology, 38(4), 470-486.

[11] Hosen, P., Himel, N., Adil, M. A., Moon, M. N. N., & Nur, M. F. N. (2019). Music Playing and Wallpaper Changing System Based on Emotion from Facial Expression. In Emerging Technologies in Data Mining and Information Security (pp. 79-87). Springer, Singapore.

[12] Kar, N., Debbarma, M. K., Saha, A., & Pal, D. R. (2012). Study of implementing automated attendance system using face recognition technique. International Journal of computer and communication engineering, 1(2), 100.

[13] Masalha, F., & Hirzallah, N. (2014). A students attendance system using QR code. International Journal of Advanced Computer Science and Applications, 5(3), 75-79.

[14] Moore, S., Armstrong, C., & Pearson, J. (2008). Lecture absenteeism among students in higher education: A valuable route to understanding student motivation. Journal of Higher Education Policy and Management, 30(1), 15-24.

[15] Moores, E., Birdi, G. K., & Higson, H. E. (2019). Determinants of university students' attendance. Educational Research, 61(4), 371-387.

[16] Newman‐Ford, L., Fitzgibbon, K., Lloyd, S., & Thomas, S. (2008). A large‐scale investigation into the relationship between attendance and attainment: a study using an innovative, electronic attendance monitoring system. Studies in Higher Education, 33(6), 699-717.

[17] Nordmann, E., Calder, C., Bishop, P., Irwin, A., & Comber, D. (2019). Turn up, tune in, don't drop out: The relationship between lecture attendance, use of lecture recordings, and achievement at different levels of study. Higher Education, 77(6), 1065-1084.

[18] Patel, A., Joseph, A., Survase, S., & Nair, R. (2019, April). Smart Student Attendance System Using QR Code. In 2nd International Conference on Advances in Science & Technology (ICAST).

[19] Pytlarz, I., Pu, S., Patel, M., & Prabhu, R. (2018). What Can We Learn from College Students' Network Transactions? Constructing Useful Features for Student Success Prediction. International Educational Data Mining Society.

[20] Romero, C., López, M. I., Luna, J. M., & Ventura, S. (2013). Predicting students' final performance from participation in on-line discussion forums. Computers & Education, 68, 458-472.

[21] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20, 53-65.

[22] Shoewu, O., & Idowu, O. A. (2012). Development of attendance management system using biometrics. The Pacific Journal of Science and Technology, 13(1), 300-307.

[23] Sloan, D., Manns, H., Mellor, A., & Jeffries, M. (2020). Factors influencing student non-attendance at formal teaching sessions. Studies in Higher Education, 45(11), 2203-2216.

[24] Terry, M., & Malik, A. (2018). Video Gaming as a Factor That Affects Academic Performance in Grade Nine. Online Submission.

[25] Varshini, A., & Indhurekha, S. (2017). Attendance system using beacon technology. International Journal of Scientific & Engineering Research, 8(5), 38.

[26] Wang, R., Chen, F., Chen, Z., Li, T., Harari, G., Tignor, S., ... & Campbell, A. T. (2014, September). StudentLife: assessing mental health, academic performance and behavioral trends of college students using smartphones. In Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing (pp. 3-14).

[27] Wang, R., Harari, G., Hao, P., Zhou, X., & Campbell, A. T. (2015, September). SmartGPA: how smartphones can assess and predict academic performance of college students. In Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing (pp. 295-306).

[28] Wang, Z., Zhu, X., Huang, J., Li, X., & Ji, Y. (2018). Prediction of Academic Achievement Based on Digital Campus. International Educational Data Mining Society.

[29] Westerman, J. W., Perez - Batres, L. A., Coffey, B. S., & Pouder, R. W. (2011). The relationship between undergraduate attendance and performance revisited: Alignment of student and instructor goals. Decision Sciences Journal of Innovative Education, 9(1), 49-67.

[30] Yang, X., Zhou, G., Taub, M., Azevedo, R., & Chi, M. (2020). Student Subtyping via EM-Inverse Reinforcement Learning. International Educational Data Mining Society.

[31] Zhou, M., Ma, M., Zhang, Y., SuiA, K., Pei, D., & Moscibroda, T. (2016, September). EDUM: classroom education measurements via large-scale WiFi networks. In Proceedings of the 2016 acm international joint conference on pervasive and ubiquitous computing (pp. 316-327).

Fair-Capacitated Clustering

Tai Le Quy
Leibniz University Hannover
Hannover, Germany
tai@l3s.de

Arjun Roy
Leibniz University Hannover
Hannover, Germany
roy@l3s.de

Gunnar Friege
Leibniz University Hannover
Hannover, Germany
friege@idmp.uni-hannover.de

Eirini Ntoutsi
Freie Universität Berlin, Germany
Berlin, Germany
eirini.ntoutsi@fu-berlin.de

ABSTRACT

Traditionally, clustering algorithms focus on partitioning the data into groups of similar instances. The similarity objective, however, is not sufficient in applications where a *fair-representation* of the groups in terms of protected attributes like gender or race, is required for each cluster. Moreover, in many applications, to make the clusters useful for the end-user, a *balanced cardinality* among the clusters is required. Our motivation comes from the education domain where studies indicate that students might learn better in diverse student groups and of course groups of similar cardinality are more practical e.g., for group assignments. To this end, we introduce the *fair-capacitated clustering problem* that partitions the data into clusters of similar instances while ensuring cluster fairness and balancing cluster cardinalities. We propose a two-step solution to the problem: i) we rely on fairlets to generate minimal sets that satisfy the fair constraint and ii) we propose two approaches, namely hierarchical clustering and partitioning-based clustering, to obtain the fair-capacitated clustering. Our experiments on three educational datasets show that our approaches deliver well-balanced clusters in terms of both fairness and cardinality while maintaining a good clustering quality.

Keywords

fair-capacitated clustering, fair clustering, capacitated clustering, fairness, learning analytics, fairlets, knapsack.

1. INTRODUCTION

Machine learning (ML) plays a crucial role in decision-making in almost all areas of our lives, including areas of high societal impact, like healthcare and education. Our work's motivation comes from the education domain where ML-based decision-making has been used in a wide variety of tasks from student dropout prediction [9], forecasting on-time graduation of students [15] to education admission decisions [21]. Recently, the issue of bias and discrimination in ML-based decision-making systems is receiving a lot of attention [28] as there are many recorded incidents of discrimination (e.g., recidivism prediction [20], grades prediction [4, 14]) caused by such systems against individuals or groups or people on the basis of *protected attributes* like gender, race etc. Bias in education is not a new problem. There is already a long literature on different sources of bias in education [24] or students' data analysis [3] as well as studies on racial bias [31] and gender bias [22]. However, ML-based decision-making systems have the potential to amplify prevalent biases or create new ones and therefore, fairness-aware ML approaches are required also for the educational domain.

In this work, we focus on *fairness in clustering*, as in educational activities, group assignments [8] and student team achievement divisions [30] are important tools that help students working together towards shared learning goals. Clustering is an effective solution for partitioning students into groups of *similar instances* [3, 26]. Traditional algorithms, however, focus solely on the similarity objective and do not consider the fairness of the resulting clusters w.r.t. protected attributes like gender. However, studies indicate that students might learn better in diverse groups, e.g., mixed-gender groups [11, 32]. Lately, fair-clustering solutions have been proposed [1, 2, 5, 6], which aim to discover clusters with a fair representation regarding some protected attributes. In this work, we adopt the cluster fairness of [6], called cluster balance, according to which protected groups must have approximately equal representation in every cluster.

In a teaching situation, it is obvious that the size of the groups should be comparable to allow a fair allocation of work among the students. As traditional clustering algorithms do not consider this requirement, clusters of varying sizes might be extracted, reducing the usefulness and applicability of the partitioning for end-users/teachers. This leads to the demand for clustering solutions that also take into account the size of the clusters. The problem is known as *capacitated clustering problem (CCP)* [25] which aims to extract clusters with a limited capacity while minimizing the total dissimilarity in the clusters. Capacitated clustering is useful in quite a few applications such as transferring goods/services from the service providers (post office, stores, etc.), garbage collection and sales force territorial de-

Tai Le Quy, Arjun Roy, Gunnar Friege and Eirini Ntoutsi "Fair-Capacitated Clustering". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 407-414. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

sign [27]. To the best of our knowledge, no solution exists that considers both fairness and capacity of clusters on the top of the similarity objective.

To this end, we propose a new problem, the so-called *fair-capacitated clustering* that ensures fairness and balanced cardinalities of the resulting clusters. We decompose the problem into two subproblems: i) the fairness-requirement compliance step that preserves fairness at a minimum threshold of balance score and ii) the capacity-requirement compliance step that ensures clusters of comparable sizes. For the first step, we generate fairlets [6], which are minimal sets that satisfy fair representation w.r.t. a protected attribute. For the second step, we propose two solutions for two different clustering types, namely hierarchical and partitioning-based clustering, that consider the capacity constraint during the merge step (hierarchical approach)/assignment step (partitioning approach). Experimental results, on three real datasets from the education domain, show that our methods result in fair and capacitated clusters while preserving the clustering quality.

2. RELATED WORK

Chierichetti et al. [6] introduced the fair clustering problem with the aim to ensure equal representation for each protected attribute, such as gender, in every cluster. In their formulation, each instance is assigned with one of two colors (red, blue). They proposed a two-phase approach: clustering all instances into fairlets - small clusters preserving the fairness measure, and then applying vanilla clustering methods on those fairlets. Subsequent studies focus on generalization and scalability. Backurs et al. [1] presented an approximate fairlet decomposition algorithm which can formulate the fairlets in nearly linear time thus tackling the efficiency bottleneck of [6]. Rösner and Schmidt [29] generalized the fair clustering problem to more than two protected attributes. A more generalized and tunable notion of fairness for clustering was introduced in Bera et al. [2]. Anshuman and Prasant [5] introduced a fair hierarchical agglomerative clustering method for multiple protected attributes.

The capacitated clustering problem (CCP), a combinatorial optimization problem, was first introduced by Mulvey and Beck [25] who proposed solutions using heuristic and subgradient algorithms. Several approaches exist to improve the efficiency of solutions or CCP approaches for different cluster types. Khuller and Sussmann [17], for example, introduced an approximation algorithm for the capacitated k-Center problem. Geetha et al. [10] improved k-Means algorithm for CCP by using a priority measure to assign points to their centroid. Lam and Mittenthal [19] proposed a heuristic hierarchical clustering method for CCP to solve the multi-depot location-routing problem.

In this work, we introduce the problem of fair-capacitated clustering which builds upon the notions from fair clustering and capacitated clustering. In particular, we build upon the notion of fairlets [6] to extract the minimal sets that preserve fairness. Regarding the CCP we follow the formulation of [25] to ensure balanced cluster cardinalities. To the best of our knowledge, the combined problem has not been studied before and as already discussed, comprises a useful tool in quite a few domains like education.

3. PROBLEM DEFINITION

Let $X \in \mathbb{R}^n$ be a set of instances to be clustered and let $d() : X \times X \to \mathbb{R}$ be the distance function. For an integer k we use $[k]$ to denote the set $\{1, 2, ..., k\}$. A *k-clustering* $\mathcal{C}$ is a partition of X into k disjoint subsets, $\mathcal{C} = \{C_1, C_2, ..., C_k\}$, called *clusters* with $S = \{s_1, s_2, ..., s_k\}$ be the corresponding cluster centers. The goal of clustering is to find an *assignment* $\phi : X \to [k]$ that minimizes the objective function:

$$\mathcal{L}(X, \mathcal{C}) = \sum_{s_i \in S} \sum_{x \in C_i} d(x, s_i) \tag{1}$$

As shown in Eq. 1, the goal is to find an assignment that minimizes the sum of distances between each point $x \in X$and its corresponding cluster center $s_i \in S$. It is clear that such an assignment optimizes the similarity but does not consider fairness or capacity of the resulting clusters.

Capacitated clustering: The goal of capacitated clustering [25] is to discover clusters of given capacities while still minimizing the distance objective $\mathcal{L}(X, \mathcal{C})$. The capacity constraint is defined as an upper bound Q_i on the cardinality of each cluster C_i:

$$|C_i| \leq Q_i \tag{2}$$

Clustering fairness: We assume the existence of a binary protected attribute $P = \{0, 1\}$, e.g., gender={male, female}. Let $\psi : X \to P$ denotes the demographic group to which the point belongs, i.e., male or female.

Fairness of a cluster is evaluated in terms of the balance score [6] as the minimum ratio between two groups.

$$balance(C_i) = \min \left(\frac{|\{x \in C_i | \psi(x) = 0\}|}{|\{x \in C_i | \psi(x) = 1\}|}, \frac{|\{x \in C_i | \psi(x) = 1\}|}{|\{x \in C_i | \psi(x) = 0\}|} \right) \tag{3}$$

Fairness of a clustering $\mathcal{C}$ equals to the balance of the least balanced cluster $C_i \in \mathcal{C}$.

$$balance(\mathcal{C}) = \min_{C_i \in C} balance(C_i) \tag{4}$$

We now introduce the problem of fair-capacitated clustering that combines all aforementioned objectives regarding distance, fairness and capacity.

Definition 1. (Fair-capacitated clustering problem)
We define the problem of (t, k, q)-fair-capacitated clustering as finding a clustering $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$ that partitions the data X into $|\mathcal{C}| = k$ clusters such that the cardinality of each cluster $C_i \in \mathcal{C}$ does not exceed a threshold q, i.e., $|C_i| \leq q$ (*the capacity constraint*), the balance of each cluster is at least t, i.e., $balance(\mathcal{C}) \geq t$ (*the fairness constraint*), and minimizes *the objective function* $\mathcal{L}(X, \mathcal{C})$. Parameters k, t, q are user defined referring to the number of clusters, minimum balance threshold and maximum cluster capacity, respectively.

4. FAIR-CAPACITATED CLUSTERING

4.1 Fairlet decomposition

Traditionally, the vanilla versions of clustering algorithms are not capable of ensuring fairness because they assign the data points to the closest center without the fairness consideration. Hence, if we could divide the original data set into subsets such that each of them satisfies the balance threshold t then grouping these subsets to generate the final

clustering would still preserve the fairness constraint. Each fair subset is defined as a fairlet. We follow the definition of fairlet decomposition by [6].

Definition 2. (Fairlet decomposition)
Suppose that $balance(X) \geq t$ with $t = f/m$ for some integers $1 \leq f \leq m$, such that the greatest common divisor $gcd(f, m) = 1$. A decomposition $\mathcal{F} = \{F_1, F_2, ..., F_l\}$ of X is a fairlet decomposition if: i) each point $x \in X$ belongs to exactly one fairlet $F_j \in \mathcal{F}$; ii) $|F_j| \leq f + m$ for each $F_j \in \mathcal{F}$, i.e., the size of each fairlet is small; and iii) for each $F_j \in \mathcal{F}$, $balance(F_j) \geq t$, i.e., the balance of each fairlet satisfies the threshold t. Each F_j is called a fairlet.

By applying fairlet decomposition on the original dataset X, we obtain a set of fairlets $\mathcal{F} = \{F_1, F_2, \ldots, F_l\}$. For each fairlet F_j we select randomly a point $r \in F_j$ as the *center*. For a point $x \in X$, we denote $\gamma : X \to [1, l]$ as the index of the mapped fairlet. The second step, is to cluster the set of fairlets $\mathcal{F}$ into k final clusters, subject to the cardinality constraint. The clustering process is described below for the hierarchical clustering type (Section 4.2) and for the partitioning-based clustering type (Section 4.3). Clustering results in an assignment from fairless to final clusters: $\delta : \mathcal{F} \to [k]$. The final fair-capacitated clustering $\mathcal{C}$ can be determined by the overall assignment function $\phi(x) = \delta(F_{\gamma(x)})$, where $\gamma(x)$ returns the index of the fairlet to which x is mapped.

4.2 Fair-capacitated hierarchical clustering

Given the set of fairlets: $\mathcal{F} = \{F_1, F_2, \ldots, F_l\}$, let $W = \{w_1, w_2, \ldots, w_l\}$ be their corresponding weights, where the weight w_j of a fairlet F_j is defined as its cardinality, i.e., number of points in F_j.

Traditional agglomerative clustering approaches merge the two closest clusters, so rely solely on similarity. We extend the merge step by also ensuring that merging does not violate the cardinality constraint w.r.t. the cardinality threshold q.

THEOREM 1. *The balance score of a cluster formed by the union of two or more fairlets, is at least t.*

$$balance(\mathcal{Y}) \geq t, \text{ where } \mathcal{Y} = \cup_{j \leq l} F_j \text{ and } balance(F_j) \geq t$$

PROOF. We use the method of induction to derive the proof. Assume we have a set of fairlets $\mathcal{F} = \{F_1, F_2, \ldots, F_l\}$, in which, $balance(F_j) \geq t, \quad j = 1, \ldots, l$. We first consider the case for any two fairlets $\{F_1, F_2\} \in \mathcal{F}$. We have $balance(F_1) = \frac{f_1}{m_1} \geq t$ and $balance(F_2) = \frac{f_2}{m_2} \geq t$. We denote by $\mathcal{Y}$ is the union of two fairlets F_1 and F_2, then

$$balance(\mathcal{Y}) = balance(F_1 \cup F_2) = \frac{f_1 + f_2}{m_1 + m_2} \tag{5}$$

It holds:

$$\begin{aligned} &\frac{f_1}{m_1} \geq t \quad \text{or, } \frac{f_1}{m_1 + m_2} \geq \frac{tm_1}{m_1 + m_2} \\ &\text{Similarly, } \frac{f_2}{m_1 + m_2} \geq \frac{tm_2}{m_1 + m_2} \\ \implies &\frac{f_1}{m_1 + m_2} + \frac{f_2}{m_1 + m_2} \geq \frac{tm_1}{m_1 + m_2} + \frac{tm_2}{m_1 + m_2} \\ \implies &\frac{f_1 + f_2}{m_1 + m_2} \geq \frac{t(m_1 + m_2)}{m_1 + m_2} = t \end{aligned} \tag{6}$$

Therefore, from Eq. 5 and Eq. 6 we get,

$$balance(\mathcal{Y}) \geq t \tag{7}$$

Thus, the statement given in Theorem 1 is true for any cluster formed by the union of any two fairlets. Now we assume that the statement holds true for a cluster formed from i fairlets, i.e, $\mathcal{Y} = \cup_{j \leq i} F_j$, where $1 < i < l$. Then,

$$balance(\mathcal{Y}) = \frac{\sum_{j \leq i} f_j}{\sum_{j \leq i} m_j} \geq t \tag{8}$$

Consider another fairlet $F_{i+1} \in \mathcal{F}$ which is not in the formed cluster $\mathcal{Y}$, $balance(F_{i+1}) = \frac{f_{i+1}}{m_{i+1}} \geq t$. Then, by joining F_{i+1} with the cluster $\mathcal{Y}$ we get the new cluster $\mathcal{Y}'$ such that,

$$balance(\mathcal{Y}') = \frac{f_{i+1} + \sum_{j \leq i} f_j}{m_{i+1} + \sum_{j \leq i} m_j} \tag{9}$$

Following the steps in Eq. 6, we can similarly show that,

$$\frac{f_{i+1} + \sum_{j \leq i} f_j}{m_{i+1} + \sum_{j \leq i} m_j} \geq t \implies balance(\mathcal{Y}') \geq t \tag{10}$$

Hence, the theorem holds true for cluster formed with $i + 1$ fairlets if it is true for i fairlets. Since i is any arbitrary number of fairlets, thus the theorem holds true for all cases. □

Theorem 1 shows that for any cluster formed by union of fairlets, the fairness constraint is always preserved. Henceforth, we don't need further interventions w.r.t. fairness.

The pseudocode is shown in Algorithm 2 of Appendix B. In each step, the closest pair of clusters is identified and a new cluster is created only if its capacity does not exceed the capacity threshold q. Otherwise, the next closest pair is investigated. The procedure continues until k clusters remain. The remaining clusters are fair and capacitated according to the correponding thresholds t and q. To compute the proximity matrix (line 1 and line 8), we use the distance between centroids of the corresponding clusters. The function *capacity*(*cluster*) in line 5 returns the size of a cluster.

4.3 Fair-capacitated partitioning clustering

Partitioning-based clustering algorithms, such as k-Medoids, can be viewed as a distance minimization problem, in which, we try to minimize the objective function in Eq. 1. The vanilla k-Medoids does not satisfy the cardinality constraint since the allocating points to clusters step is only based on the distance among them. Now, if we change the goal of this assignment step to find the "best" data points with a defined capacity for each medoid instead of searching for the most suitable medoid for each point, we can control the cardinality of clusters. We formulate the problem of *assigning points to clusters* subject to a capacity threshold q as a knapsack problem [23].

Let $S = \{s_1, s_2, ..., s_k\}$ be the cluster centers, i.e., medoids, and $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ be the resulting clusters. We change the assignments of points to clusters, using knapsack, in order to meet the capacity constraint q. In particular, we define a flag variable $y_j = 1$ if x_j is assigned to cluster C_i, otherwise $y_j = 0$. Now, we define a value v_j to data point x_j, which depends on the distance of x_j to C_i, with v_j being

maximum if C_i is the best cluster for x_j, i.e, the distance between x_j and s_i is minimum. We formulate value v_j of instance x_j based on an exponential decay distance function:

$$v_j = e^{-\frac{1}{\lambda} * d(x_j, s_i)} \tag{11}$$

where $d(x_j, s_i)$ is the Euclidean distance between the point x_j and the medoid s_i. The higher λ is the lower the effect of distance in the value of the points. The point which is closer to the medoid will have a higher value.

Then, the objective function for the assignment step is:

$$\text{maximize} \sum_{j=1}^{n} v_j y_j \tag{12}$$

Now, given $\mathcal{F} = \{F_1, F_2, \ldots, F_l\}$ and $W = \{w_1, w_2, \ldots, w_l\}$ are the set of fairlets and their corresponding weights, i.e., the number of instances in the fairlet, respectively; q is the maximum capacity of the final clusters. Our target is to cluster the set of fairlets $\mathcal{F}$ into k clusters centered by k medoids. We apply the formulas in Eq. 11 and Eq.12 on the set of fairlets $\mathcal{F}$, i.e, each fairlet F_j has the same role as x_j. Then, the problem of assigning the fairlets to each *medoid* in the cluster assignment step becomes finding a set of fairlets with total weights less than or equal to q and the total value is maximized. In other words, we can formulate the cluster assignment step in the partitioning-based clustering as a **0-1 knapsack problem**.

$$\begin{aligned} &\text{maximize} \sum_{j=1}^{l} v_j y_j \\ &\text{subject to} \quad \sum_{j=1}^{l} w_j y_j \leq q \quad \text{and} \quad y_j \in \{0, 1\} \end{aligned} \tag{13}$$

In which, y_j is the flag variable for F_j, $y_j = 1$ if F_j is assigned to a cluster, otherwise $y_j = 0$; v_j is the value of F_j which is computed by the Eq. 11; q is the desired maximum capacity.

The pseudocode of our k-Medoids fair-capacitated algorithm is described in Algorithm 2. In which, for each *medoid* we would search for the adequate points (line 3) by using function $knapsack(p, values, w, q)$ (line 10) implemented using dynamic programming, which returns a list of items with a maximum total value and the total weight not exceeding q. In the main function, line 12, we optimize the clustering cost by replacing *medoids* with *non-medoid* instances when the clustering cost is decreased. This optimization procedure will stop when there is no improvement in the clustering cost is found (lines 19 to 32).

Algorithm 1: k-Medoids fair-capacitated algorithm

Input: $\mathcal{F} = \{F_1, F_2, \ldots, F_l\}$: a set of fairlets
$W = \{w_1, w_2, \ldots, w_l\}$: weights of fairlets
q: a given maximum capacity of final clusters
k: number of clusters
Output: A fair-capacitated clustering

```
1  Function ClusterAssignment(medoids):
2      clusters ← ∅;
3      for each medoid s in medoids do
4          candidates ← all fairlets which are not assigned
             to any cluster ;
5          p ← length(candidates) ;
6          w ← weights(candidates) ;
7          for each fairlet_i in candidates do
8              values[i] ← v(fairlet_i) //Eq. 11 ;
9          end
10         clusters[s]←knapsack(p, values, w, q) ;
11     end
12     return clusters;
13 Function main():
14     medoids ← select k of the l fairlets arbitrarily ;
15     ClusterAssignment(medoids) ;
16     cost_best ← current clustering cost;
17     s_best ← null ;
18     o_best ← null ;
19     repeat
20         for each medoid s in medoids do
21             for each non-medoid o in F do
22                 consider the swap of s and o, compute
                     the current clustering cost;
23                 if current clustering cost < cost_best then
24                     s_best ← s;
25                     o_best ← o;
26                     cost_best ← current clustering cost;
27                 end
28             end
29         end
30         update medoids by the swap of s_best and o_best ;
31         ClusterAssignment(medoids)
32     until no improvements can be achieved by any
         replacement;
33 return clusters;
```

5. EXPERIMENTS

In this section, we describe our experiments and the performance of our proposed methods on three educational datasets.

5.1 Experimental setup

Datasets. The datasets are summarized in Table 1.

UCI student performance. This dataset includes the demographics, grades, social and school-related features of students in secondary education of two Portuguese schools [7] in 2005 - 2006. "Gender" is selected as the protected attribute, i.e., we aim to balance gender in the resulting clusters.

Open University Learning Analytics (OULAD). This is the dataset from the OU Analyse project [18] of Open University, England in 2013 - 2014. Information of students includes their demographics, courses, their interactions with the virtual learning environment, and final outcome. We aim to balance the "Gender" attribute in the results.

MOOC. The data covers students who enrolled in the 16 edX courses offered by the two institutions (Harvard University and the Massachusetts Institute of Technology) during 2012 - 2013 [13]. The dataset contains aggregated records which represent students' activities and their final grades of the courses. "Gender" is the protected attribute.

Baselines. We compare against well-known fairness-aware

Table 1: An overview of the datasets

Dataset	#instances	#attributes	Protected attribute	Balance score
UCI student performance	649	33	Gender (F: 383; M: 266)	0.695
OULAD	4,000	12	Gender (F: 2,000; M: 2,000)	1
MOOC	4,000	21	Gender (F: 2,000; M: 2,000)	1

clustering methods and a vanilla clustering method.

k-Medoids [16]: a traditional partitioning clustering technique that divides the dataset into k clusters so as to minimize clustering cost. Cluster centers are actual instances.

Vanilla fairlet [6]: a fairness-aware clustering approach that i) decomposes the dataset into fairlets and ii) applies a vanilla k-center clustering algorithm [12] to form the final k clusters.

MCF fairlet [6]: Similar to Vanilla fairlet but the fairlet decomposition part is transformed into a *minimum cost flow* (MCF) problem, by which an optimized version of fairlet decomposition in terms of cost value is computed.

Evaluation measures. We report on clustering quality (measured as clustering cost, see Eq. 1), cluster fairness (expressed as cluster balance, see Eq. 4) and cluster capacity (expressed as cluster cardinality).

Parameter selection. Regarding fairness, a minimum threshold of balance t is set to 0.5 for all datasets in our experiments. It means that the proportion of the minority group is at least 50% in each resulting cluster. Regarding the λ factor in Eq. 11, a value $\lambda = 0.3$ is chosen for our experiments from a range of [0.1, 1.0] via grid-search. We evaluate the clustering cost and balance score on a small dataset, i.e., UCI student performance dataset - Mathematics subject w.r.t λ. Theoretically, the **ideal capacity** of clusters is $\left\lceil \frac{|X|}{k} \right\rceil$ where $|X|$ is the population of dataset X, k is the number of desired clusters. However, in many cases, the clustering models cannot satisfy this constraint, especially the hierarchical clustering model. Hence, we set the formula in Eq. 14 to compute the **maximum capacity** q of clusters; ε is a parameter chosen in experiments for each fair-capacitated clustering approach.

$$q = \left\lceil \frac{|X| * \varepsilon}{k} \right\rceil \tag{14}$$

To find the appropriate value of ε, we set a range of [1.0, 1.3] to ensure all the generated clusters have members and evaluate the cardinality of resulting clusters on the UCI student performance (Mathematics subject) dataset. ε is set to 1.01 and 1.2, for k-Medoids fair-capacitated and hierarchical fair-capacitated methods, respectively.

5.2 Experimental results

UCI student performance. When k is less than 4, as shown in Figure 1-a, the clustering quality of our models can be close to that of the vanilla k-Medoids method. However, the clustering cost is fluctuated thereafter due to the effort to maintain the fairness and cardinality of methods. Our *vanilla fairlet hierarchical fair-capacitated* outperforms other competitors in most cases. Vanilla fairlet and MCF fairlet show the worst clustering cost as an effect of the k-Center method. Figure 1-b depicts the clustering fairness. As we can observe, in terms of fairness, *vanilla fairlet hierarchical fair-capacitated* has the best performance when k is less than 10. Contrary to that, by selecting each point for each cluster in the cluster assignment step, the *k-Medoids fair-capacitated* method can maintain well the fairness in many cases. Regarding the cardinality, as illustrated in Figure 1-c, our approaches outperform the competitors when they can keep the number of instances for each cluster under the specified thresholds.

OULAD. Our *MCF fairlet k-Medoids fair-capacitated* approach outperforms other methods in terms of clustering cost, although there is an increase compared to the vanilla k-Medoids algorithm, as we can see in Figure 2-a. Concerning fairness, in Figure 2-b, k-Medoids is the weakest method while others can achieve the highest balance. The balance of *Gender* feature in the dataset is the main reason for this result. All fairlets are fully fair; this is a prerequisite for our methods of being able to maintain the perfect balance. Regarding cardinality, our approaches demonstrate their strength in ensuring the capacity of clusters (Figure 2-c). The difference in the size of the clusters generated by our methods is tiny. This is in stark contrast to the trend of competitors.

MOOC. The results of clustering quality are described in Figure 3-a (Appendix A). Although an increase in the clustering cost is the main trend, our methods outperform the vanilla fairlet and MCF fairlets methods. Regarding clustering fairness, as depicted in Figure 3-b, our approaches can maintain the perfect balance for all experiments. This is the result of an actual balance in the dataset and the fairlets. The emphasis is our methods can divide all the experimented instances into capacitated clusters, as shown in Figure 3-c, which proves their superiority in presenting the results over the competitors regarding clusters' cardinality.

Summary of the results. In general, fairness is well maintained in all of our experiments. When the data is fair, in case of OULAD and MOOC datasets, our methods achieve a perfect fairness. In terms of cardinality, our methods are able to maintain the cardinality of resulting clusters within the maximum capacity threshold, which is significantly superior to competitive methods. The fair-capacitated partitioning based method is better than the hierarchical approach since it can determine the capacity threshold closest to the *ideal capacity*. Regarding the clustering cost, the hierarchical approach has an advantage over other methods by outperforming its competitors in most experiments.

6. CONCLUSION AND OUTLOOK

In this work, we introduced the fair-capacitated clustering problem that extends traditional clustering, solely focusing on similarity, by also aiming at a balanced cardinality among the clusters and a fair-representation of instances in each cluster according to some protected attributes like gender or race. Our solutions work on the fairlets derived from

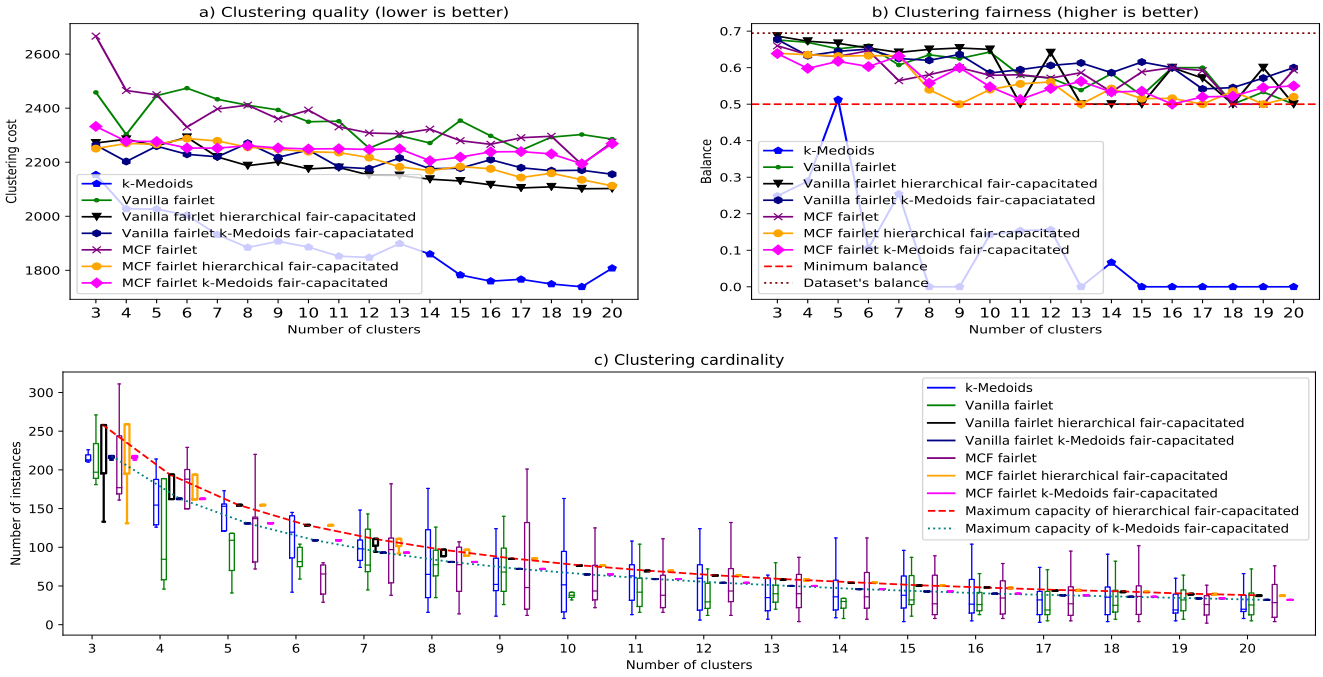


Figure 1: Performance of different methods on UCI student performance dataset

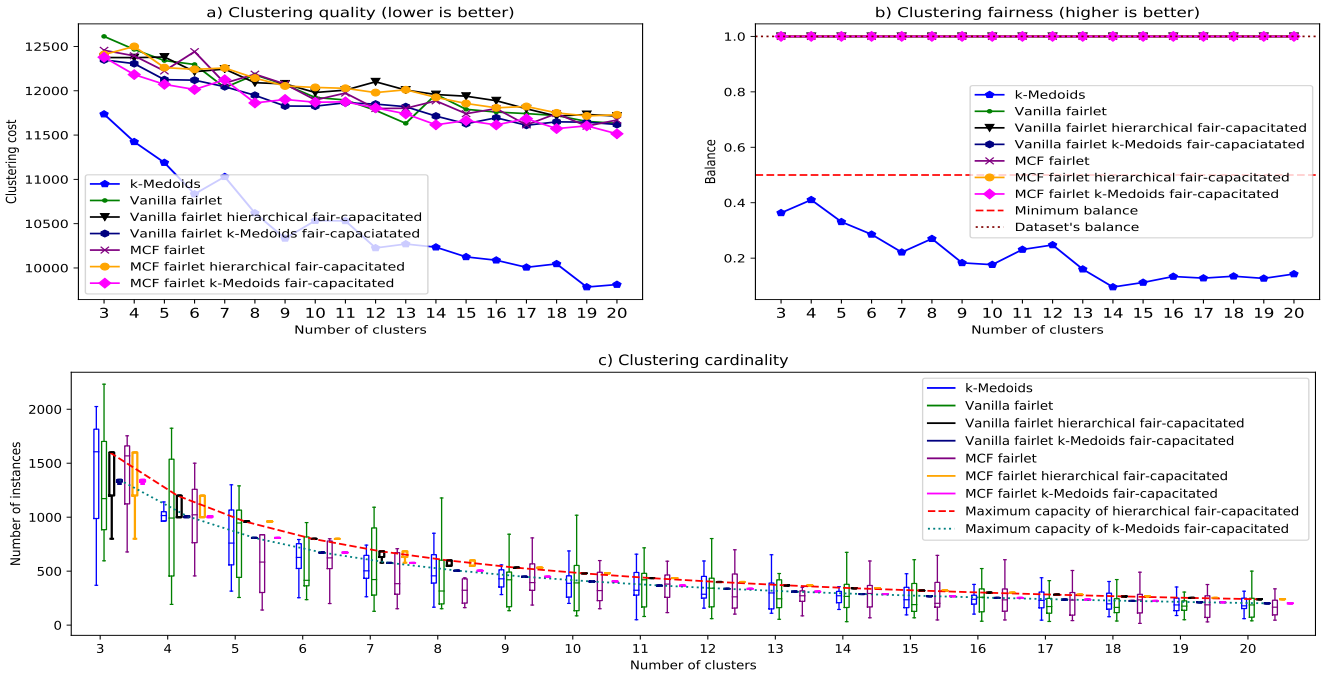


Figure 2: Performance of different methods on OULAD dataset

the original instances: the hierarchical-based approach takes into account the cardinality requirement during the merging step, whereas the partitioning-based approach takes into account the cardinality of the final clusters during the assignment step which is formulated as a knapsack problem. Our experiments show that our methods are effective in terms of fairness and cardinality while maintaining clustering quality. In the future, we plan to extend our approach for more than one protected attributes as well as to further investigate what fair group assignments means in educational settings.

Acknowledgements

The work of the first author is supported by the Ministry of Science and Education of Lower Saxony, Germany, within the PhD program "LernMINT: Data-assisted teaching in the MINT subjects". The work of the second author is supported by the Volkswagen Foundation under the call "Artificial Intelligence and the Society of the Future".

7. REFERENCES

[1] A. Backurs, P. Indyk, K. Onak, B. Schieber, A. Vakilian, and T. Wagner. Scalable fair clustering. In *International Conference on Machine Learning*, pages 405–413. PMLR, 2019.

[2] S. Bera, D. Chakrabarty, N. Flores, and M. Negahbani. Fair algorithms for clustering. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[3] S. Bharara, S. Sabitha, and A. Bansal. Application of learning analytics using clustering data mining for students' disposition analysis. *Education and Information Technologies*, 23(2):957–984, 2018.

[4] K. Bhopal and M. Myers. The impact of covid-19 on a level students in england. *SocArXiv*, 2020.

[5] A. Chhabra and P. Mohapatra. Fair algorithms for hierarchical agglomerative clustering. *arXiv preprint arXiv:2005.03197*, 2020.

[6] F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii. Fair clustering through fairlets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5036–5044, 2017.

[7] P. Cortez and A. M. G. Silva. Using data mining to predict secondary school student performance. *EUROSIS-ETI*, 2008.

[8] M. Ford and J. Morice. How fair are group assignments? a survey of students and faculty and a modest proposal. *Journal of Information Technology Education: Research*, 2(1):367–378, 2003.

[9] J. Gardner, C. Brooks, and R. Baker. Evaluating the fairness of predictive student models through slicing analysis. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, pages 225–234, 2019.

[10] S. Geetha, G. Poonthalir, and P. Vanathi. Improved k-means algorithm for capacitated clustering problem. *INFOCOMP*, 8(4):52–59, 2009.

[11] D. Gnesdilow, A. Evenstone, J. Rutledge, S. Sullivan, and S. Puntambekar. Group work in the science classroom: How gender composition may affect individual performance. 2013.

[12] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

[13] HarvardX. HarvardX Person-Course Academic Year 2013 De-Identified dataset, version 3.0, 2014.

[14] S. Hubble and P. Bolton. A level results in england and the impact on university admissions in 2020-21. *House of Commons Library*, 2020.

[15] S. Hutt, M. Gardner, A. L. Duckworth, and S. K. D'Mello. Evaluating fairness and generalizability in models predicting on-time graduation from college applications. *International Educational Data Mining Society*, 2019.

[16] L. Kaufman and P. J. Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, 344:68–125, 1990.

[17] S. Khuller and Y. J. Sussmann. The capacitated k-center problem. *SIAM Journal on Discrete Mathematics*, 13(3):403–418, 2000.

[18] J. Kuzilek, M. Hlosta, and Z. Zdrahal. Open university learning analytics dataset. *Scientific data*, 4:170171, 2017.

[19] M. Lam and J. Mittenthal. Capacitated hierarchical clustering heuristic for multi depot location-routing problems. *Int. J. Logist. Res. Appl.*, 16(5):433–444, 2013.

[20] J. Larson, S. Mattu, L. Kirchner, and J. Angwin. How we analyzed the compas recidivism algorithm. *ProPublica (5 2016)*, 9(1), 2016.

[21] F. Marcinkowski, K. Kieslich, C. Starke, and M. Lünich. Implications of ai (un-) fairness in higher education admissions: the effects of perceived ai (un-) fairness on exit, voice and organizational reputation. In *FAT* '20*, pages 122–130, 2020.

[22] T. Masterson. An empirical analysis of gender bias in education spending in paraguay. *World Development*, 40(3):583–593, 2012.

[23] G. B. Mathews. On the partition of numbers. *Proceedings of the London Mathematical Society*, 1(1):486–490, 1896.

[24] M. Meaney and T. Fikes. Early-adopter iteration bias and research-praxis bias in the learning analytics ecosystem. In *Companion Proceeding of the 9th International Conference on Learning Analytics & Knowledge (LAK'19), Fairness and Equity in Learning Analytics Systems Workshop*, pages 14–20, 2019.

[25] J. M. Mulvey and M. P. Beck. Solving capacitated clustering problems. *European Journal of Operational Research*, 18(3):339–348, 1984.

[26] Á. A. M. Navarro and P. M. Ger. Comparison of clustering algorithms for learning analytics with educational datasets. *IJIMAI*, 5(2):9–16, 2018.

[27] M. Negreiros and A. Palhano. The capacitated centred clustering problem. *Computers & operations research*, 33(6):1639–1663, 2006.

[28] E. Ntoutsi, P. Fafalios, U. Gadiraju, V. Iosifidis, W. Nejdl, M.-E. Vidal, S. Ruggieri, F. Turini, S. Papadopoulos, E. Krasanakis, et al. Bias in data-driven artificial intelligence systems-an introductory survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(3):e1356, 2020.

[29] C. Rösner and M. Schmidt. Privacy preserving clustering with constraints. *arXiv preprint arXiv:1802.02497*, 2018.

[30] M. Tiantong and S. Teemuangsai. Student team achievement divisions (stad) technique through the moodle to enhance learning achievement. *International Education Studies*, 6(4):85–92, 2013.

[31] N. Warikoo, S. Sinclair, J. Fei, and D. Jacoby-Senghor. Examining racial bias in education: A new approach. *Educational Researcher*, 45(9):508–514, 2016.

[32] Z. Zhan, P. S. Fong, H. Mei, and T. Liang. Effects of gender grouping on students' group performance, individual achievements and attitudes in computer-supported collaborative learning. *Computers in Human Behavior*, 48:587–596, 2015.

APPENDIX

A. MOOC DATASET

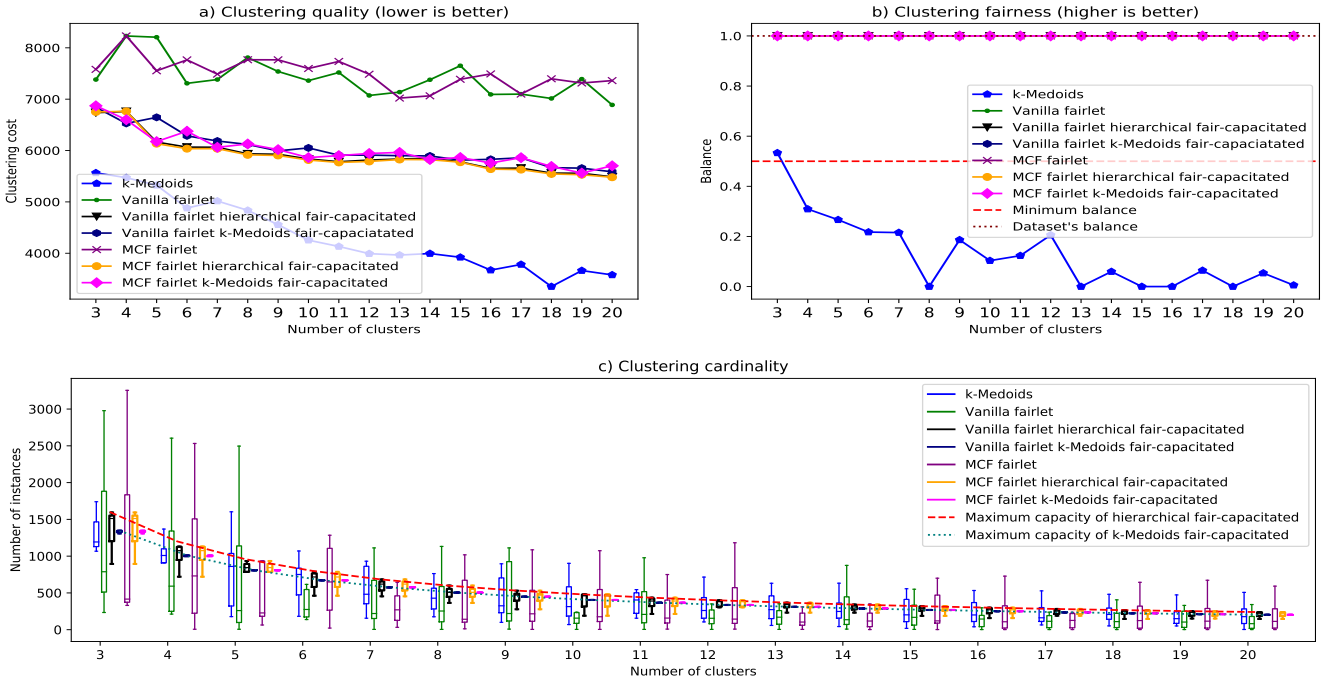


Figure 3: Performance of different methods on MOOC dataset

B. HIERARCHICAL FAIR-CAPACITATED ALGORITHM

Algorithm 2: Hierarchical fair-capacitated algorithm

Input: $\mathcal{F} = \{F_1, F_2, \ldots, F_l\}$: a set of fairlets
q: a given maximum capacity of final clusters
$W = \{w_1, w_2, \ldots, w_l\}$: weights of fairlets
k: number of clusters

Output: A fair-capacitated clustering

compute the proximity matrix ;
$clusters \leftarrow \mathcal{F}$ //each fairlet F_j is considered as cluster ;
repeat
  $cluster_1, cluster_2 \leftarrow$ the closest pair of clusters ;
  if $capacity(cluster_1) + capacity(cluster_2) \leq q$ **then**
    $newcluster \leftarrow \text{merge}(cluster_1, cluster_2)$;
    update $clusters$ with $newcluster$;
    update the proximity matrix ;
  else
    continue;
  end
until k *clusters remain*;
return *clusters*;

Combining Cognitive and Machine Learning Models to Mine CPR Training Histories for Personalized Predictions

Florian Sense
InfiniteTactics, LLC
Dayton, OH, USA
florian.sense@infinitetactics.com

Michael Krusmark
L3Harris Technologies
Melbourne, FL, USA
michael.krusmark.ctr@us.af.mil

Joshua Fiechter
Ball Aerospace
Dayton, OH, USA
jfiechte@ball.com

Michael G. Collins
ORISE at AFRL
Dayton, OH, USA
michael.collins.74.ctr@us.af.mil

Lauren Sanderson & Joshua Onia
RQI Partners, LLC
Dallas, TX, USA
lauren.sanderson@rqipartners.com
josh.onia@rqipartners.com

Tiffany Jastrzembski
Air Force Research Laboratory
Dayton, OH, USA
tiffany.jastrzembski@us.af.mil

ABSTRACT

Cardiopulmonary resuscitation (CPR) is a foundational life-saving skill for which medical personnel are expected to be proficient. Frequent refresher training is needed to prevent the involved skills from decaying. Regular low-dose, high-frequency training for staff at fixed intervals has proven successful at maintaining CPR competence but does not take into account inherent performance variability across learners. Tailoring refreshers to an individual's past performance would minimize personnel being trained too (in)frequently and would ensure faster knowledge acquisition for new learners. To maximize the benefits of individualized schedules, learning needs gleaned from past training history need to be identified. A recent field study conducted among nursing students showed that a cognitive model-based approach was able to successfully trace the knowledge acquisition and decay of learners and prescriptively devise personalized training regimes that outperformed fixed schedules with regards to both training efficiency and learners' performance. Here, we report a post-hoc analysis of the collected data to investigate whether an alternative modeling approach, blending cognitive modeling and machine learning, could have resulted in even higher quality predictions. Our results reveal modest improvements in predictive accuracy for ensemble models, in which machine learning models predict the prediction errors (i.e., residuals) of the standalone cognitive model. These promising findings reveal strong applied utility for future use in domains where sustained proficiency is a requirement.

Keywords

Predictive modeling; Cognitive model; Machine learning; Cardiopulmonary resuscitation; Learning

Florian Sense, Michael Krusmark, Joshua Fiechter, Michael G. Collins, Lauren Sanderson, Joshua Onia and Tiffany Jastrzembski "Combining Cognitive and Machine Learning Models to Mine CPR Training Histories for Personalized Predictions". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 415-421. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

1. INTRODUCTION

Cardiopulmonary resuscitation (CPR) is a basic life-saving skill but it has been shown that medical professionals are not able to perform it consistently [1]. To remedy this shortfall, several improvements to skill acquisition and maintenance programs have been proposed [6]. One dimension of the shift in educational focus [5] emphasizes increased re(training) efficiency by moving towards personalized, adaptive scheduling. The current work aims to facilitate this development.

Currently, as in many domains, refresher trainings at fixed intervals (i.e., regular and the same for everyone) are required to maintain CPR compliance. A recent effort [14, 18] has shown that a cognitive model representing regularities of memory can be leveraged to devise personalized training schedules that maintain proficiency at lower cost and risk. This effort will be referred to as *the CPR field study* throughout the current text, and the data collected during this experiment (see sections 2.1 and 2.2 for details) will form the bedrock on which the efforts presented here will build.

Specifically, we conducted a post-hoc simulation study of the CPR field study data to explore whether the cognitive model's predictions could be enhanced by combining it with machine learning (ML) models that can leverage additional information to improve predictive accuracy. The combination of the two modeling approaches is achieved by fitting the models sequentially, forming an ensemble model in which the cognitive model's residuals are learned by the ML models. We show that their combined predictions afford a modest improvement over the cognitive model by itself and are preferable to using the ML models by themselves.

Modern CPR training is an interesting educational data mining domain and modeling task because trainings are conducted on advanced manikins equipped with an array of sensors that quantify various aspects of a student's performance against objective performance guidelines [16]. Consequently, large amounts of high-resolution data are readily collected for a given event. The challenge is that events are usually spaced months apart, which provides a sparse sampling space for knowledge tracing. Consequently, it is difficult to make precise predictions. However, given quality CPR's central

role in the "chain of survival" [6, 5], even small improvements in predictive accuracy can conceivably have large real-life impacts—especially if predictive models can identify those most in need of more frequent refresher trainings and help them to maintain compliance.

Generally speaking, the fields of cognitive science and machine learning have approached the computational modeling of a task like CPR skill acquisition and maintenance with different mindsets [24]. Specifically, cognitive models primarily focus on explaining the mechanisms that drive individual differences; machine learning models primarily focus on out-of-sample prediction. Aiming to combine the best of both approaches, we engineered a pipeline of predictive models: First, a cognitive model that was specifically developed as a prescriptive tool [13] is fit to the training data, which results in residuals that indicate which instances are fit poorly by the cognitive model. Next, a ML model is fit to those residuals, effectively learning to fine-tune the cognitive model's predictions. We show that such an ensemble approach can provide improvements in predictive accuracy without sacrificing interpretability, which is important to retain so that the model's personalized prescriptions are fully explainable. With an eye on advancing predictive tools in the domain of CPR training, our core motivation is to assess whether alternative predictive approaches could have yielded better result in the CPR field study, so that insights gained can be leveraged in future applied settings.

1.1 The current study

Here, we will use the exact version of the cognitive model that was used in the CPR field study as a yardstick to determine: **(I)** How well would alternative instantiations of the cognitive model have performed?, **(II)** Would a number of off-the-shelf ML approaches have yielded superior predictions than the cognitive model?, and **(III)** Could ML models be used to learn the prediction error of the cognitive model?

We believe the last question is the most pertinent. However, the combined approach should be compared to the approaches that only use either of the two modeling approaches in isolation to ascertain whether it has any benefit.

2. METHODS

This section will outline the data we used for our exploration, the input features that are available in the data, the predictive model we employed, and the setup of the simulation study we conducted to address our research questions. Figure 1 provides a schematic overview of the approach and its parts and connections are going to be explained in the following.

2.1 Data

Data were from a multi-phased, longitudinal study conducted at 10 nursing schools across the United States [18]. A total of 475 nursing students started the study. Participants were randomly assigned to 4 initial acquisition conditions where they completed 4 consecutive CPR training sessions that were spaced by 1 day, 1 week, 1 month, or 3 months. Students were additionally randomized to 3 maintenance training conditions where they refreshed their skills for 1 year at intervals of 3 months, 6 months, or at personalized intervals prescribed by the cognitive model. During each session, students completed a series of CPR events using the Resuscitation Quality Improvement (RQI®) system with Laerdal's Resusci Anne® adult QCPR manikin.

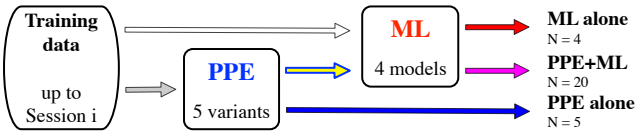


Figure 1: Schematic of the various predictive models.

First, students completed a pre-test consisting of 60 compressions or 12 bag-mask ventilations with no feedback from the manikin, followed by trainings where students received real-time, dynamic feedback to guide, and then post-tests with no feedback. RQI provides composite scores for the quality of compressions and ventilations on scales of 0 to 100, with higher scores corresponding to better performance. The compression score is based on depth, rate, release, and hand placement. The ventilation score is based on volume, rate, and compliance with inspiration time.

Prior to the onset of the study, participants completed a demographic questionnaire. Of the 475 participants that began the study, we included in our study the 393 that completed the initial acquisition phase. Due to the variations in retraining schedules across the maintenance phase, not all students completed the same number of sessions. We focus here on data from the first through the eighth session since the majority of students completed 8 sessions.

2.2 Input features

This section describes all information available in the "training data" box of Figure 1. As sessions progress, the number of available training instances grows but the number of input features is constant. Using the color-coded arrows in Figure 1 to categorize them, these features are detailed in the following and the `labels` correspond to those in Figure 4A.

Gray arrow in Figure 1: `time/lag`: An event's timestamp expressed as the number of seconds since the first/previous recorded event; `score`: The composite performance score recorded for each event.

White arrow in Figure 1: `compvent`: Does the recorded event correspond to performing compressions or ventilations?; `acqint` and `maintint`: What acquisition-maintenance interval combination was this user assigned to? Together, these define the 12 experimental conditions (see previous section for detail) that the condition PPE variant is based on; `session`: Session counter 1 through 8; `pretrnpst`: Was this a pre-test, training, or post-test?; `site`, `age`, `gender`, `height`, and `weight`: Demographic information associated with each user (time-invariant). There were ten sites/locations and other information was coded in years, male/female, inches, and pounds, respectively; `profile`: We reduced the unique user IDs to a low-dimensional set of performance profiles. This approach was inspired by earlier work [2, 23] that showed a small number of descriptive profiles could be obtained by performing k-means clustering [9]. Here, we used $k = 4$, and re-partitioned the training data on each iteration of the

simulation. Specifically, only pre-test scores across both skills were taken into account.

Yellow arrow in Figure 1: `decay rate` and `model time`: The two PPE terms computed when fitting PPE (see section 2.3.1) are passed to the ML models; `residual`: The difference between PPE's fit and `score`.

The three rightmost arrows indicate the predictions that are made, emphasizing that the PPE+ML ensemble models (purple) uses both the cognitive (blue) and machine learning (red) models. Next, we outline which ML models were used and how the five PPE variants were fit to the data.

2.3 Predictive models

As noted in Figure 1, there are a total of 29 models. Here, we describe the PPE and ML models that make up the PPE alone/ML alone predictions. The remaining majority of models are based on combining the PPE+ML models by first fitting the PPE as described below and subsequently computing the PPE's residuals in the training data and training the ML model to predict those. The PPE+ML predictions can thus be thought of PPE predictions that were fine-tuned by a given ML model.

2.3.1 Predictive performance equation (PPE)

The PPE is a set of nested mathematical equations that capture findings in the cognitive science literature associated with the temporal dynamics of human learning and forgetting [26]. These include the power law of learning, the power law of forgetting, the spacing effect, and effects of relearning. Two essential components of PPE are sub-equations that model time and the rate of knowledge/skill decay. The *model time* equation captures the idea that the age of items in memory should be skewed toward the most recent presentations, but the full study history should be represented. Hence, model time for each instance i (across n instances) is $w_i \times t_i$, where $w_i = \frac{t_i^{-0.6}}{\sum_{j=1}^{n} t_j^{-0.6}}$ and t_i is the time, in seconds, relative to the first instance. The *decay rate* equation captures the idea that spacing practice across time produces more stable knowledge that decays at a slower rate, while massing practice produces less stable knowledge that decays at a faster rate. Since model time and the decay rate are essential to how PPE captures learning and decay, we include them separately as features in the machine learning models. For a more extensive description of the mechanics of the PPE, please see [25, 26].

In the CPR field study, PPE was fit separately to each participant's history of compression and ventilation scores. We refer to this variant as the *original* PPE throughout the paper. The rationale for individual fits was from prior research suggesting that each individual would have unique learning and forgetting trajectories across sessions due to psychometric differences, the trajectories would vary for compressions and ventilations, and thus predictive accuracy would be maximized by fitting to each student on each skill.

Here, we conduct post-hoc simulations to explore these assumptions by comparing the methodology used in the field study to less granular PPE variants in which free parameters are fit to: experimental condition (acquisition and maintenance intervals), skill (compressions and ventilations), user, or user's performance profile. By exploring these different groupings for which a set of unique parameters are estimated, we evaluate the trade-space between model flexibility and predictive accuracy, and how this interacts with the amount of data available for fitting the models.

2.3.2 Machine learning models

We used four different machine learning models. Depending on the approach, these were either trained to predict the score (red arrow in Figure 1) or PPE's residuals (purple arrow in Figure 1). For either task, all models had access to all input features outlined in section 2.2 (also see x-axis of Figure 4A). All models were run with the default settings of the cited `R` packages [21].

As the simplest model, we fit a single **decision tree** to the scores/residuals. Each tree was pruned through 10-fold cross validation—as implemented in [22]—to avoid overfitting. In most cases, this resulted in very shallow trees and sometimes even single node "stumps." Hence, the decision trees can be thought of as baseline models.

The most complex model was a **random forest** [4], which is an ensemble of decision trees. Using the default settings in [15], we used both bagging and random feature sub-setting to grow a forest of 500 trees. A recent comparison of gradient boosting algorithms included random forests as a comparison and nicely showed that they perform very well on a range of ML tasks and have the added benefit of not requiring hyperparameter tuning [3]. The disadvantage of random forests—as with many ML approaches—is that the internal mechanics that result in a prediction are not easily inspected (see our discussion around Figure 4 below).

The two other models were **ridge** regression and the **lasso** [10], which apply slightly different shrinkage terms when coefficients are estimated. The key difference between the two methods is that ridge regression will retain coefficients for all input features, while the lasso effectively performs feature selection (see section 6.2 in [12] for an introduction). This generally makes lasso models more interpretable. Both models have a single hyperparameter, λ, that was tuned for each iterative prediction using the cross-validation procedure implemented in [8] and all numerical features were standardized.

2.4 Simulation study and analysis

The approach to our simulation study can be summarized as follows: For each session $s = 1, 2, \ldots, 7$, train the models on data up to session s and issue predictions for the pre-test of session $s + 1$. We focused on pre-test scores because we were interested in predicting students' readiness to perform CPR compressions and ventilations, prior to additional training. The procedure was run for all 29 (combinations of) models described above and generated iterative predictions for sessions 2 through 8. The quality of predictions across the models will be compared by computing the mean absolute error (MAE), which summarizes how accurately, on average, each model predicts the scores recorded in the subsequent session. This yields $7 \times (4 + 20 + 5) = 203$ (i.e., number of predicted sessions times number of models) errors. For the sake of easier presentation of these results, we subsequently

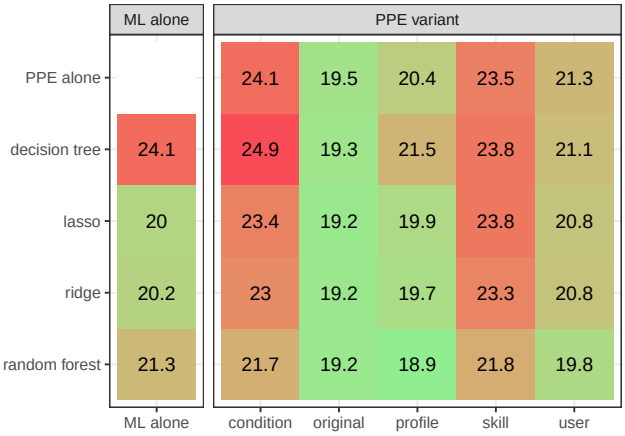


Figure 2: Average MAE across sessions for all models.

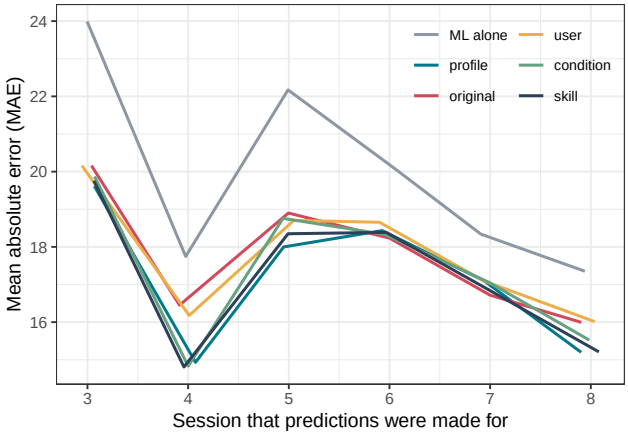


Figure 3: Comparison of all models in the "random forest" row of Figure 2, showing prediction errors for each session.

summarize the errors by (i) computing the average MAE for each model (Figure 2), and (ii) ranking the models based on their errors (Table 2). These overall results are elaborated on with additional figures and tables that highlight relevant details.

3. RESULTS

Figure 2 presents the average MAE for all 29 models and speaks to all three research questions posed in section 1.1. As detailed in section 2.4, the 203 prediction errors were aggregated across sessions and the average MAE for each combination of models is presented as a heatmap. The color-coding corresponds to the magnitude of the errors; lower values are better. By averaging across sessions, variations in predictive accuracy as a function of session (see Figure 3) is lost but it becomes easier to assess the model's relative performance in one glance.

First, we can look at the "PPE alone" row in Figure 2 to compare the five PPE variants that were tested. Overall, the original instantiation of the cognitive model as used in the CPR field study—if used alone—does indeed outperform the more constrained variants explored here. This is somewhat surprising since the original model exhibited signs of overfitting (i.e., fit MAE lower than prediction MAE) that were ameliorated for the constrained variants of PPE. However, it appears that despite overfitting the training data, the original variant of PPE did produce the best predictions after all.

Second, whether a number of off-the-shelf ML approaches would have yielded superior predictions than the cognitive model can be assessed by comparing the cell original PPE alone (MAE = 19.5) with the prediction errors in the "ML alone" column in Figure 2. All ML approaches yield average errors larger than 19.5 when applied alone, which suggests that the ML models tested here—if used by themselves—would not have resulted in better predictions overall. However, the differences in prediction errors are not large and the ridge and lasso regression in particular perform well on average.

And third, and most importantly, we turn to the PPE+ML combinations. These correspond to the larger facet labeled "PPE variants" in Figure 2. A number of notable patterns emerge: the average MAE for the original PPE is hardly affected by adding any of the ML models to predict its residuals. This might be because this variant of PPE is very flexible, which restricts the residuals in the training data that the ML can actually fit to. For all other PPE variants, we see a gradient from top to bottom, with average MAEs decreasing with ML models relative to PPE alone. The decision trees are an exception to this pattern and seem to worsen the performance more often than not. Otherwise, we generally see the lasso and ridge regressions improving on PPE alone and the PPE+random forest resulting in the best performance for all PPE variants.

Zooming in on the models using random forests, Figure 3 shows the session-by-session prediction errors of the random forest alone (ML alone) and the PPE+random forest combinations. We omitted predictions for the second session because they are quite poor for the random forests combined with the condition and skill PPE variants, which distorts the y-axis and obscures differences between models in the later sessions[1]. The figure highlights that the random forest alone consistently performs worse than all other combinations of models, in which the random forest is used specifically to learn PPE's residuals rather than observed performance. This suggests that the most promising approach is an ensemble of a PPE variant that captures the overall temporal dynamics to issue predictions that are subsequently fine-tuned by a random forest that can leverage all other available input features.

Another way to summarize these results is by ranking all 29 models' MAE within each session and computing the average rank for each PPE variant. These average ranks are shown

[1]Predictions for session 2 were generally much worse than for all subsequent sessions. We ran all analyses reported here without session 2 predictions to confirm that our conclusions do not depend on differences between models on session 2. If session 2 is omitted, the skill PPE variant performs a little better overall but results are not otherwise affected drastically.

Table 1: Comparison of PPE variants across all ML models.

PPE variant	average rank	average MAE
profile	11.9	20.1
skill	12.7	23.2
original	15.1	19.3
user	17.2	20.8
condition	18.4	23.4

in Table 1, and reveal that although the original PPE yields the lowest overall average MAE, both the profile and skill variants achieve better average ranks. This suggests that if ML models are leveraged to predict PPE's residuals, more constrained variants of PPE tend to perform better. However, even the lowest average ranks listed in Table 1 is relatively high, suggesting that no model consistently outperforms the others. This observation is confirmed by inspecting the models' MAEs in detail (not shown here), which reveals that for some sessions, most models perform effectively identically.

Lastly, we present the top 10 models in terms of their ranking in Table 2. Here, the ranks are computed as an average across the seven sessions each model made predictions for. The best-performing model is the PPE with parameters for each performance profile whose residuals are predicted by a random forest. Figure 2 corroborates this observations, showing that this combination of models obtains the lowest average MAE overall. Notably, all five instances of the original PPE and five out of six instances of random forests are represented in the top 10, confirming that these models perform very well in various combinations.

3.1 Peeking into the random forest

Space constraints limit the amount of model interrogation we can report here. However, we want to at least showcase one prominent example. Table 2 and Figure 2 show that the best model overall is the combination of the PPE variant with unique parameters for each performance `profile` and a random forest that learns its residuals. (This model is the blue line in Figure 3, which highlights that other models perform very similarly.) Figure 4A shows the normalized feature importance computed for each input feature (white and yellow arrows in Figure 1) for each iterative session that predictions are made for. Superimposed are the average importance and the spread (in black) and features are sorted from least to most important based on average importance. Notably, most time-invariant features (gender, age, etc.) are equally important across the seven iterations. The `session` counter, `stability`, and `model time`, on the other hand, become gradually more important as more sessions were included in the training data, while the opposite pattern is evident for `compvent` and users' performance `profile`.

Feature importance plots as shown here can be informative but should be interpreted with caution since they do not capture and visualize the potential intricate non-linear relationships between the various input features [17]. Furthermore, feature importance and their impact on predictions are not necessarily the same—more advanced approaches exist [19] but are beyond the scope of the current paper.

Table 2: The top 10 models overall sorted by average rank across the seven predictions made by each model.

	PPE variant	ML model	average rank
1	profile	random forest	5.3
2	skill	random forest	6.7
3	condition	random forest	7.9
4	original	lasso	8.1
5	original	random forest	8.3
6	original	decision tree	8.4
7	original	ridge	8.6
8	user	random forest	10.1
9	original	PPE alone	10.7
10	condition	ridge	11.9

Figure 4B zooms in on two important features and shows the predictions made for the profile PPE model for the fourth session against the residuals the random forest predicts for each instance. We generally see the most differentiation between models on Session 4, which is why we chose it—however, this figure is broadly representative of the profile PPE+RF dynamics for other sessions. Figure 4B suggests that ventilations are more often down-adjusted than compressions (i.e., more triangles below the equality line) unless PPE predicts near-ceiling performance. The fact that model time is consistently identified as the most important feature (see Figure 4A) but no clear relationship between the magnitude of the adjustment (i.e., distance from equality line) emerges in Figure 4B highlights the disadvantage of applying ML models—such as a random forest—that are challenging to interrogate.

4. DISCUSSION

The post-hoc simulation study reported here suggests that the original cognitive model used for prescriptive, adaptive scheduling in the CPR field study performed very well overall. In fact, in the aggregate, it resulted in lower average prediction errors than both the more constrained variants of PPE and the machine learning models included in the current comparison. Thus, it is unlikely that the tested off-the-shelf ML models would have performed better than the original PPE, although the regularized regression models (ridge and lasso) in particular achieved prediction errors similar to the original PPE. We expected the ML models to outperform the cognitive model because the latter's main "insights" (the estimated *model time* and *decay rate*) were included as input features to the ML models (yellow arrow in Figure 1. This suggests that the PPE, using much less information, was slightly better at extrapolating performance to the next session.

The current explorations also showed, however, that an ensemble cognitive and ML model has the potential to perform slightly better than either alone. Notably, the more constrained variants of PPE performed particularly well in this ensemble arrangement. One possible explanation is that the less flexible cognitive model operates as a smoothing function on the temporal information, which leaves the ML to learn under which conditions (i.e., [combinations of] input features) the general temporal dynamics should be adjusted to further improve predictions. This framing of the procedure is

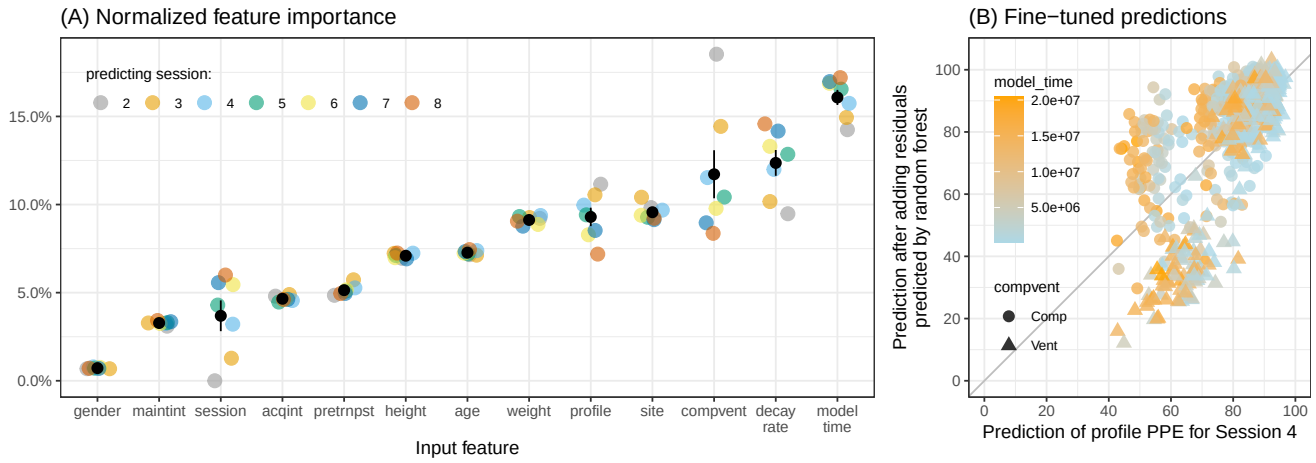


Figure 4: Details on the best-performing model. (A): The normalized feature importance for each iteration of the model with superimposed averages (black dots). (B): Initial PPE predictions for Session 3 (x-axis) plotted against fine-tuned predictions (y-axis); color-coding indicates the `model time` for each predicted instance and shape differentiates `compvent`.

conceptually akin to a two-step boosting algorithm [7] and the "fine-tuning" of predictions induced by the second step (the ML model in this case) is nicely illustrated in Figure 4B, which shows the results of the first step (the constrained cognitive model's predictions on the x-axis), against the results after the second boosting-like step (the PPE+ML predictions on the y-axis). In this process, the initial PPE prediction's quite restricted range is expanded by the random forest, which can—and does (cf. Figure 4A)—draw on various input features.

The ensemble approach outlined here has the added benefit of a modular structure. Thus, it is easy to make design decisions, particularly regarding (i) which constraints should be built into the parameter fitting procedure for PPE, and (ii) what type of ML model is most informative. The latter will determine where on the continuum of interpretability the ensemble will fall. For example, the dynamics of the random forest that predicts the profile PPE's residuals (highlighted in Figure 4), does not lend itself to straightforward model interrogation but the PPE+lasso and PPE+ridge combinations would not reduce the interpretability of the ensemble, while slightly reducing prediction errors relative to PPE alone (see Figure 2).

It should be pointed out, however, that the improvement in prediction errors relative to the original PPE alone is minor. Nevertheless, we consider these findings significant for two reasons: First, the small improvement vindicates that PPE's time-based mechanisms capture the majority of variance in this task domain. Second, the PPE+ML ensemble approach used here serves as a proof-of-concept that illustrates how the core mechanism of PPE can be preserved while incorporating an arbitrary number of additional input features. For example, some of the input features used here were specific to the field study's design (notably `acqint` and `maintint`) and would not be present in the hospital setting RQI systems are primarily deployed in. In such settings, however, other input features would be available (e.g., job title or department) and samples would be larger and more heterogeneous, which would conceivably introduce more variance that is not a function of time-based features alone. We expect that under these conditions, the ensemble approach's advantage over PPE alone would be more pronounced.

In the current effort, we choose to assess the models' ability to make session-by-session predictions. This approach meant that events did not line up chronologically (a student in the weekly acquisition condition will have completed the first four session before a student in the 3-month condition returned for their second session) but the amount of training data available for each student is equalized—only the lag between events varies. This reveals, for example, that predictions improve up to session 4 (the end of the acquisition phase; see Figure 3) and then get worse for session 5, which is when students switch to the maintenance phase. This suggests that the models get better at forecasting performance as more data from a consistent schedule becomes available, and that one should expect a dip in predictive accuracy as the temporal dynamics are altered.

Future work in this domain should validate the approach presented here in more naturalistic data that more closely resemble how medical professionals train and maintain CPR proficiency. We believe that cognitive models in particular—and a cognitive-machine learning ensemble specifically—hold great promise in moving towards a predictive framework that affords personalized, adaptive refresher training schedules that are tailored towards individual learning needs—either of an individual or groups of learners that exhibit similar performance profiles. Furthermore, the outlined predictive pipeline's potential value in adaptive, educational learning system outside of the medical domain should be explored.

5. ACKNOWLEDGEMENTS

This work was funded through the 711th Human Performance Wing Chief Scientist Seedling award at the Air Force Research Laboratory. Data were wrangled in `R` using [28]; tables were created with [11], and figures with [27] and [20].

6. REFERENCES

[1] B. S. Abella, J. P. Alvarado, H. Myklebust, D. P. Edelson, A. Barry, N. O'Hearn, T. L. V. Hoek, and L. B. Becker. Quality of cardiopulmonary resuscitation during in-hospital cardiac arrest. *Jama*, 293(3):305–310, 2005.

[2] E. Ayers, R. Nugent, and N. Dean. Skill set profile clustering based on student capability vectors computed from online tutoring data. *Educational Data Mining*, 2008.

[3] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, pages 1–31, 2020.

[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] A. Cheng, D. J. Magid, M. Auerbach, F. Bhanji, B. L. Bigham, A. L. Blewer, K. N. Dainty, E. Diederich, Y. Lin, M. Leary, et al. Part 6: resuscitation education science: 2020 american heart association guidelines for cardiopulmonary resuscitation and emergency cardiovascular care. *Circulation*, 142(16_Suppl_2):S551–S579, 2020.

[6] A. Cheng, V. M. Nadkarni, M. B. Mancini, E. A. Hunt, E. H. Sinz, R. M. Merchant, A. Donoghue, J. P. Duff, W. Eppich, M. Auerbach, et al. Resuscitation education science: educational strategies to improve outcomes from cardiac arrest: a scientific statement from the american heart association. *Circulation*, 138(6):e82–e122, 2018.

[7] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[8] J. H. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software, Articles*, 33(1):1–22, 2010.

[9] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):100–108, 1979.

[10] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[11] M. Hlavac. *stargazer: Well-Formatted Regression and Summary Statistics Tables*. Central European Labour Studies Institute (CELSI), Bratislava, Slovakia, 2018. R package version 5.2.2.

[12] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*. Springer, 2013.

[13] T. S. Jastrzembski, K. A. Gluck, and G. Gunzelmann. Knowledge tracing and prediction of future trainee performance. In *Interservice/Industry Training, Simulation, and Education Conference*, pages 1498–1508. National Training Systems Association, 2006.

[14] T. S. Jastrzembski, M. Walsh, M. Krusmark, S. Kardong-Edgren, M. Oermann, K. Dufour, T. Millwater, K. A. Gluck, G. Gunzelmann, J. Harris, et al. Personalizing training to acquire and sustain competence through use of a cognitive model. In *International conference on augmented cognition*, pages 148–161. Springer, 2017.

[15] A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2(3):18–22, 2002.

[16] R. M. Merchant, A. A. Topjian, A. R. Panchal, A. Cheng, K. Aziz, K. M. Berg, E. J. Lavonas, D. J. Magid, A. Basic, P. B. Advanced Life Support, R. E. S. Advanced Life Support, Neonatal Life Support, and S. of Care Writing Groups. Part 1: Executive summary: 2020 american heart association guidelines for cardiopulmonary resuscitation and emergency cardiovascular care. *Circulation*, 142(16_Suppl_2):S337–S357, 2020.

[17] C. Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

[18] M. Oermann, M. Krusmark, S. Kardong-Edgren, T. S. Jastrzembski, and K. A. Gluck. Personalized training schedules for retention and sustainment of CPR skills. *Simulation in Healthcare*, 2021.

[19] T. Parr, J. D. Wilson, and J. Hamrick. Nonparametric feature impact and importance. *arXiv preprint arXiv:2006.04750*, 2020.

[20] T. L. Pedersen. *patchwork: The Composer of Plots*, 2019. R package version 1.0.0.

[21] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[22] B. Ripley. *tree: Classification and Regression Trees*, 2019. R package version 1.0-40.

[23] F. Sense, M. Collins, M. Krusmark, and T. S. Jastrzembski. Using k-means clustering for out-of-sample predictions of memory retention. In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society*, 2020.

[24] G. Shmueli et al. To explain or to predict? *Statistical science*, 25(3):289–310, 2010.

[25] M. M. Walsh, K. A. Gluck, G. Gunzelmann, T. Jastrzembski, and M. Krusmark. Evaluating the theoretic adequacy and applied potential of computational models of the spacing effect. *Cognitive science*, 42:644–691, 2018.

[26] M. M. Walsh, K. A. Gluck, G. Gunzelmann, T. Jastrzembski, M. Krusmark, J. I. Myung, M. A. Pitt, and R. Zhou. Mechanisms underlying the spacing effect in learning: A comparison of three computational models. *Journal of Experimental Psychology: General*, 147(9):1325, 2018.

[27] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[28] H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019.

Math Multiple Choice Question Solving and Distractor Generation with Attentional GRU Networks

Neisarg Dave*
Pennsylvania State University
nud83@psu.edu

Riley Bakes*
Pennsylvania State University
rob5372@psu.edu

Barton Pursel
Pennsylvania State University
bkp10@psu.edu

C. Lee Giles
Pennsylvania State University
clg20@psu.edu

ABSTRACT

We investigate encoder-decoder GRU networks with attention mechanism for solving a diverse array of elementary math problems with mathematical symbolic structures. We quantitatively measure performances of recurrent models on a given question type using a test set of unseen problems with a binary scoring and partial credit system. From our findings, we propose the use of encoder-decoder recurrent neural networks for the generation of mathematical multiple-choice question distractors. We introduce a computationally inexpensive decoding schema called character offsetting, which qualitatively and quantitatively shows promise for doing so for several question types. Character offsetting involves freezing the hidden state and top k probabilities of a decoder's initial probability outputs given the input of an encoder, then performing k basic greedy decodings given each of the frozen outputs as the initialization for decoded sequence.

Keywords

Math Question Solving, Distractor Generation, Math Multiple Choice Questions, Mathematical Language, Math Education

1. INTRODUCTION

1.1 Problem Statement

Here we focus on the needs of mathematics educators in high school and early university education, One of the most tedious jobs for a teacher is to create exams and quizzes and grade them. The more time they spend on these tasks, the less time they spend teaching students. An automated system capable of creating reliable math questions of consistent difficulty level, creating solutions, generating distractors for them, and finally be able to grade them is the holy grail of educational automation. In this paper we focus on solving the math questions and generating distractors for Multiple Choice Questions.

Questions in mathematics are different from other subjects such as English, History, or Economics. In mathematics and by extension in all STEM fields, questions and answers not only are composed of natural text but are often accompanied by symbolic equations, expressions, inequalities or relational information. Ganeslingam [6] postulates in his work that these non-textual objects not only augment the context of the textual part but also derive their context from it. These non-textual objects are not part of the natural language and hence require special treatment. On the semantic level, mathematical questions require an underlying understanding of rules before a question can be solved. A mere comprehension is not sufficient. To solve a simple problem in arithmetic, the fundamental understanding of the four operators is necessary.

In this work we experiment with a network which has had historical success on natural language processing problems and test its ability to generalize mathematical knowledge from an open source data set contributed by [22] consisting of elementary focused question types. Alternatively as a second problem, for some question types, we examine whether models which fail to generalize to the test set may have their incorrect solutions leveraged as 'good' distractor options for multiple choice questions like those seen in multiple choice questions on a math quiz. As following with the precedent of the data set contributor [22], mathematical expressions are presented using Python's operator syntax.

In summary we show the following:

- Insight in the ability of an encoder decoder attentional GRU to extract semantic and syntactic meaning from mathematical expressions.
- Simultaneously test whether these model's incorrect predictions may be leveraged to auto generate multiple choice question distractors commonly seen in lower education exams. Continuing with this potential application, experiment with the practice of character offsetting–a modified greedy decoding schema which pushes the networks to predict four separate sequences instead of a single output thus providing a complete set of distractors.

*Both authors contributed equally.

Neisarg Dave, Riley Owen Bakes, Bart Pursel and C. Lee Giles "Math Multiple Choice Question Solving and Distractor Generation with Attentional GRU Networks". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 422-430. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

- Qualitatively measure which question types show the highest potential for leveraging character offsetting for the purpose of multiple choice distractor generation.

2. RELATED WORK

Math Word Problems

Early applications of machine learning and deep learning based methods in math questions attempted to convert math word problems into equations [10]. Here we rely on the capability of neural network to identify the textual and equation parts from the question and model them correctly in order to solve them. Much of the work in math word problems relied on extracting equations from text and then solving them using symbolic solver libraries such as Sympy.

Subhro Roy and Dan Roth created expansion trees [20] and Unit Dependency Graphs [21] from arithmetic word problems. Kushman et. al. [11] mapped word problems to equations using canonical templates and handled ambiguity using probabilistic models. Tencent AI Lab [29] first used deep neural networks for solving math word problems. They used the Seq2Seq model with LSTM units for mapping math word problems to the equations. MathDQN [27] proposed using Deep Q-Learning to map math word problems to solvable equations. Deepmind [15] attempted to solve math word problems using the program induction technique, which would also generate a rationale for choosing an answer. This method did not involve mapping problems to equations but the reasoning in text form. Recent work focused on using recursive neural networks for evaluating equations [30] [28] mapped from word problems.

Polozov et. al. [19] and Liu et. al [16] proposed methods for generating math word problems. Liu et. al. [16] used Gated Graph Neural Networks with Variational Autoencoders to generate questions from knowledge graphs of mathematical concepts and symbolic expressions. [19] take programming approach for encoding requirements from tutors and students to create logical graphs with the help of ontology. This logical graph is then used to create expressions and sentences with the help of primitive templates.

Lample et. al. [12] showed solutions for questions in differential equations, differential calculus, and integral calculus and used transformer networks [26] to solve calculus questions and compared their results with traditional solvers like Mathematica and Matlab. In contrast, Saxton et. al. [22] created a codebase to generate math problems across fifty-six classes and solve them using deep neural networks. Saxton et. al. compared problem solving abilities of Seq2seq networks with transformers. Though transformers showed better results than the the recurrent models, but Saxton et. al. commented that the improvement in performance was largely due to the higher capacity of transformer networks to remember rather than their ability to solve.

Here we use the codebase created by Saxton et. al. [22] to generate math questions. Even though the codebase in its original form cannot generate distractors, it can be modified to create distractors using simple rules. In comparison to other datasets [15] which contain distractors, we chose to use the codebase since it provided more control over the generation of questions and also the templates used have a simpler language and an equation with each question.

In a classroom and tutoring settings math questions are more open ended. Erikson et. al [5] tested the capability of XGBoost, Random Forests and LSTMs in analyzing the open ended answers in mathematics. These models were created to assist the teacher rather than complete automated grading. Michalenko et. al [17] used LSTMs to solve polynomial factorization problems. They created their dataset from Wolfram Alpha. They use the trained network for automated grading and personalized feedback system.

Distractor Generation (DG)

In multiple choice questions, the options which are not the answer are called distractors, because their job is to distract a student from a given correct answer. Distractor generation has been studied for non-mathematical subjects, especially English (Susanti et. al. [23]) and other domain-specific tasks (Aldabe et. al. [2]). Distractor generation for scientific subjects like physics, chemistry, biology, and economics was explored by Linag et. al. [13]. They [13] used a two-stage model with a classifier and a ranker to filter out the relevant distractors. Linag et. al. [14] explored distractors for fill in the blank type questions using GAN networks.

Partial Credit Scoring

Similar in spirit to our partial credit scoring system, Pho et. al. [18] attempt to automatically score the *quality* of manually created English multiple choice distractors using various semantic and syntactic criteria including WordNet. This is fundamentally different from our problem however as we seek to automate the generation of the distractors themselves and simultaneously sought out a metric to help measure the fundamental reasonableness of those distractors.

3. EXPERIMENTS

3.1 Training Data

The data set used in this paper [22] had the express purpose of being a large scale training and testing framework for benchmarking models on mathematical reasoning. The framework consists of both training and testing sets. The training set consists of 39 different math problem types and variants of 17 of those add the element of mathematical composition to the problem's statements for a total of 56 question types organized into 8 different domains–probability, polynomials, numbers, measurement, comparison, calculus, arithmetic, and algebra. Each question type within a domain is split into three training sets, easy, medium and hard of 666,666 question answer pairs, for a total of 2 million examples per question type.

Difficulty measures the relative complexity of coefficients in the expressions generated. As an example compare from the polynomial evaluation set the easy: 'Let u(q) = q**2 - 6*q - 10. Calculate u(8).', medium: 'Let s(v) = v**3 + 47*v**2 +471*v + 142. Give s(-33)', hard: 'Let h(a) = -177071*a - 4957992. What is h(-28)?' and an actual related college algebra exam question [25]: 'Evaluate the function f(x) = 3 + (x-5)**(1/2) at x = 9.'. It is relatively clear that

examples from medium and hard are unlikely to appear on a low level math examination. For this reason the majority of experiments rely on the easy train set variants. It was our hypothesis that as the curriculum provided by these sets are much more in line with the expected complexity of questions appearing on actual low level exams that the models would thus be more likely to generate better distractors (or even the correct answer) when provided such an exam's questions as input.

The primary test method within the framework proposed by [22] is a data set referred to as interpolation. Every question type has an associated interpolation test set. The set consists of 10^5 question answer pairs *likely* unseen in either the easy, medium and hard train sets of the associated question. The guarantee of lack of repeated questions comes from a lower bound on the probability of a questions repeated generation. A training question at most has a 10^{-8} chance of reappearing in the test set. [22] also release a secondary test set referred to as extrapolation, which measures generalization of core concepts across multiple question types. However, as we specifically were interested in single question trained models for the express purpose of multiple choice question distractor generation this test set was unused.

3.2 Rule Based Distractor Generation

Evaluation of distractors is not an exact process. For a given question there can be any number of distractors, some good, other bad. There is usually a very loose concensus on what constitutes a good distractor. Also no algorithm exists that can gauge the "goodness" of distractor. However, expert educators have a keen sense of judging the distractor by their teaching and research experience. Educators usually know where students make mistakes, and leverage that to generate good distractors. For simple high school questions, we can simulate this process by creating rules that mimic the mistakes made by students. These rules then can be embedded with a mathematical solver to produce distractors for given question. A simple set of rules can be created to modify the solution steps to generate distractors for questions containing mathematical equality or inequality. Commonly used rules are:

- *Change One Sign* : Randomly pick one coefficient or constant in equality/ inequality and multiply by -1.
- *Change Two Signs* : Randomly pick two coefficients or constants of equality/inequality and multiply by -1.
- *Most Frequent Number* : Use the most frequent number in the equality/inequality as a distractor
- *Nearest Multiple* : Randomly pick a coefficient or a constant in equality/inequality and change it to the nearest multiple of 2, 3 or 5.
- *Random Drop* : Randomly drop one of the coefficients or constant in equality/inequality
- *Invert Range* : Invert the solution range of the inequality, e.g. change $[0, 1]$ to $(-\infty, 0) \cup (1, \infty)$
- *Trivial Solution* : For inequality problems, one of the distractors can be chosen from $\{\phi\}$, $(-\infty, \infty)$, or *No Solution*. For equations, choices are from 0, -1 or 1
- *Flip brackets* : Change an open bracket in answer to closed and vice and versa. In the question, order of operations can be changed by changing the position of brackets.

These rules can be coded as python functions and then selected one or two rules at random to modify the steps involved in solving the question. Symbolic library like sympy can be used for generating and solving the math questions. The library developed by deepmind [22] can create math questions across various domains with varying difficulty level. We modify their codebase to extend its capability to generate the distractors based on the stated rules. Table 1 shows few examples of rule based distractor generation.

Question	Answer	Distractors
Let $-\frac{2p}{3} - \frac{2}{3} \geq \frac{2p}{5} - \frac{4}{5}$. What is p?	$-\infty < p \leq \frac{1}{8}$	$\frac{11}{2} \leq p < \infty$ $3 \leq p < \infty$ $\frac{4}{23} \leq p < \infty$
Find all solutions to $\frac{3}{2} - \frac{w}{6} \geq -\frac{2w}{11} - \frac{14}{11}$.	$-183 \leq w < \infty$	$-\frac{61}{4} \leq w < \infty$ $-\infty < w \leq \frac{61}{4}$ $-\infty < w \leq \frac{47}{2}$
Solve the polynomial inequality: $51 - 3f \neq -f - \frac{1}{6}$	$f \neq \frac{307}{12}$	$f \neq -\frac{305}{24}$ $f \neq -\frac{305}{18}$ $f \neq -\frac{46}{3}$

Table 1: Distractors generated using rules

Distractors generated using rules act as a form of reference distractors. For qualitative evaluation of distractors generated by neural networks, we will look at both the distractors side by side in table 3.

3.3 Experiment Detail

Two principle experiments can be identified. An attentional [3] encoder decoder GRU [4] is trained on a single question type for the entire 666,666 easy train set. Keeping with the spirit of the framework released by [22] we after training a model on a specific question set test the models on their respective question's interpolate test rather than a subset of the train set.

Simultaneously, during the second round of data collection with the GRU, when the model is scored on the interpolate set we perform character offsetting (see 3.3.1) and ask the model to predict 3 distractors in addition to a primary solution sequence. Two different scores were calculated for a model's performance on the test set–the first, a complete binary accuracy where credit is assigned if and only if the entire primary greedy decoded sequence matches the true solution sequence. And second, a partial credit score which is calculated by subtracting from 1 the normalized Levenshtein edit distance between the predicted and true solution. Normalized in this context means the ratio of the edit distance to the max sequence length of either the true or predicted solution sequence. Thus for a given Levenshtein distance d for solution sequence S and prediction sequence P we have partial credit defined as

$$\text{partial credit} = 1 - \frac{d(P, S)}{\max(\text{len}(P, S))} \tag{1}$$

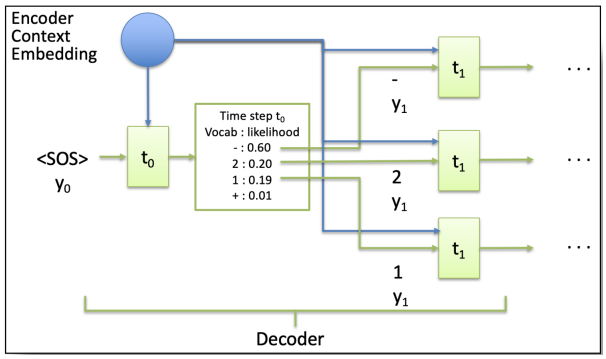


Figure 1: Example of character offsetting given an encoder's context embedding from the input question '-2 + 1'. Note decoder hidden states not shown in diagram. <SOS> signifies the start of sequence character.

It is important to acknowledge the widely different possible interpretations for what partial credit could mean in this context and why we choose the definition we did. In a typical educational setting, partial credit is assigned when the student demonstrates sufficient understanding of the problem albeit fail to arrive at the final correct solution. This is difficult to measure in model outputs–consider a correct sequence being -2 and a predicted sequence of 2, the partial credit score would be $1 - \frac{1}{2} = 50\%$. In real life a teacher may realize a student failed to report a final sign change and assign significant credit. Such thorough review is impossible given the $100,000$ examples within a single question type's test set (and also impossible considering the model's inability to provide secondary and tertiary decision making steps)–and so we formulate the above definition as an attempt to empirically measure the ability for a model to predict *similar* expressions to the correct one. We emphasize this is not any attempt to measure the models comprehension of the mathematics it is predicting on. As for why this is important; the ultimate goal is to generate multiple choice question distractors which generally exhibit some form of similarity to the correct solution. We discuss defining a good distractor more fully in section 3.3.2.

3.3.1 Character Offsetting

We propose a modified greedy decoding schema called character offsetting for generating multiple choice question distractors. In a typical greedy decoding scheme for an encoder decoder sequence to sequence model an input sequence (in our case, a string literal representation of a math problem) is given to the encoder which generates a context embedding [24]. Then character by character the decoder outputs a response sequence based on this context. At every time step of the output sequence's prediction, the previously predicted character and hidden states, as well as the encoder's context output, are re-fed into the decoder. The actual output of the decoder at any step is a probability array the size of the model's vocabulary [8]. The highest value corresponds to the most likely next character in the sequence, at least according to the model's weightings. In a greedy decoding at every step we simply take the most probable character and append it to the final output. Prediction is halted once the model outputs the end of sequence character as the most probable next step [8].

In character offsetting we freeze the initial decoder returned probability array and hidden states. We now ask the decoder to generate four total prediction sequences–one being your expected greedily decoded output, a second with the second most likely character from the frozen initial probability array as the sequence's starting character, and similarly a third and fourth. Each time a new sequence is attempted, we reset the hidden state to the saved initial hidden tensor. This was found for several question types to generate diverse and reasonable incorrect outputs. Table 4 provides a qualitative ranking based on question type for this task.

3.3.2 Difficulty in Defining a Good Distractor

Defining a good distractor is a non-trivial endeavor, and we make no claim to have accomplished this in this paper. Rather we discuss *qualities* typically considered when trying to formulate distractors for a multiple choice assessment.

Some qualities are readily apparent–general reasonability of a distractor as a possible solution to the question posed is perhaps the most fundamental requirement [7]. Measuring reasonability may be accomplished in several ways. Difference in value between a distractor and the true solution are potentially a good baseline–a distractor should be within a context specific similar magnitude as the true solution to avoid immediate exclusion. For lower level maths such as the the problem types discussed we believe this to be typically within a magnitude difference.

3.4 Model Parameters and Training Procedure

The model experimented with was an encoder decoder attentional GRU trained on a single NVIDIA 1080TI GPU for a single train-easy curriculum question type from the [22] data set. The models encoder and decoders had an embedding layer of size 512, with the decoder having 16 attentional heads. Initially what was attempted for a given training question type was an encoder decoder hidden size of 2048 on a batch size of 256. If the 1080TI GPU memory was insufficient given a training question type then we alternated between dropping encoder size and batch size. The parameters for a given question type are recorded in table 4. 150 training epochs were performed.

We follow most of the parameters used in [22]–the Adam optimizer [9] was selected for minimizing the sum of the log probabilities of the correct character with learning rate $lr = 6 * 10^{-4}$, and $\beta_1 = 0.9$, $\beta_2 = 0.995$, and $\epsilon = 10^{-9}$ and absolute gradient clipping of 0.1. The model leveraged teacher forcing during training and used 0.9/0.1 split of training data into a train and validation set.

4. RESULTS AND ANALYSIS

4.1 Attentional GRU on the Interpolate Set

4.1.1 Performance Considerations

It should be noted that these models have removed from them the greater context provided by the train-medium and train-hard data sets, of which interpolate attempts to test understanding for as well. It is possible as well that the hard or medium sets better generalize to interpolate for specific question sets. A small test seems to support this–we let the model train on the hard variant of algebra_linear_1d and scores improved from 3.9% to 44.3%.

Metric	Mean Score
Binary	0.065
Partial Credit	0.679

Table 2: Comparison of attentional GRU partial credit and binary scoring performance averaged across all questions not disqualified in 4.2.1. This includes low potential questions excluded in table 4.

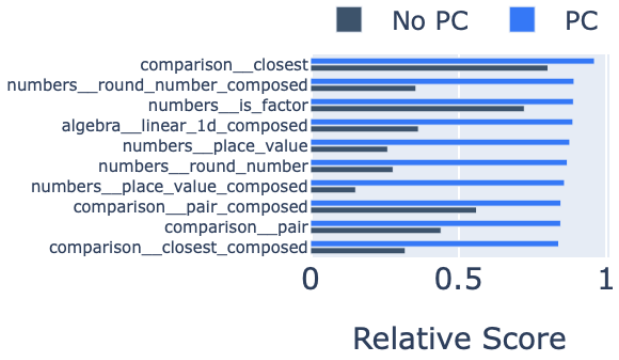


Figure 2: Top ten scoring questions when evaluated with the partial credit (PC) metric and no partial credit (binary) (No PC) score.

4.1.2 Analysis: GRU Performance Binary Scoring

Performance when scored with no partial credit varied widely. Of the top ten scoring question types 5 are from the comparison module type. Based on overall poor performance we are skeptical that the models extrapolate true mathematical semantic meaning–rather they likely just determine meta solution strategies. For example, one hypothesis for the comparison task success is that the notion of magnitude is readily apparent based on the length of the input sequence. Two observations about this–the first is that this requires the model to have gained the ability to isolate critical numeric subsequences within a larger question prompt. Secondly, providing contrary evidence to this hypothesis of sequence length metagaming, is that even for examples comparing long decimal sequences of lesser magnitude to short whole integers of greater magnitude we observe successful predictions. Lastly, the implication of signage in comparison was understood well by the models.

Generally it would appear that magnitude is an easier concept for statistical pattern recognition to abstract. Most difficult for the network was the evaluation of polynomials and other algebraic expressions, and arithmetic. Binary scores in all these categories were low–even given the considerations provided in section 4.1.1.

4.1.3 GRU Performance Partial Credit Scoring

Measurement with the partial credit metric (table 2, figure 2) demonstrates greater consistency and performance and shows promise for the ability of the attentional GRU to capture the essence of a reasonable response and work as a distractor generator for multiple choice questions–especially for types of problems the model performed poorly on using a binary scoring metric. However we note some flaws in the measurement. An example: in algebra_linear_2d_composed the model would frequently predict a single negative sign, a safe prediction and given the length of correct outputs is typically only one or two characters this led to a significant boost in score while answers remained meaningless. Interestingly in the question set's non-composed variant algebra_linear_2d the model's outputs are mostly meaningful and the partial credit score seemingly justified. To supplement the partial credit score, a qualitative examination of the reasonability of model's outputs compared to their empirical partial credit scoring is provided in table 4.

4.2 Multiple Choice Question DG

4.2.1 Considerations

The [22] framework releases a wide range of question types posed in diverse formats. Not discussed until now is that several formats are not conducive towards training models. Take for example questions from the comparison_closest set which are themselves posed as multiple choice questions–

'Which is the nearest to -955? (a) -3/4 (b) 0.2 (c) 17/3 (d) 3/5 (e) 0.5'

A model is supposed to predict either a, b, c, d, or e. Of the data sets released only four were found to use such a format for some or all questions within the set. A similar problem; six sets were either partially or fully posed as simple true and false questions. Naturally such questions are removed from consideration from our goal of distractor generation.

Partial credit was found to be an effective indication for many question types of whether a model's principle predicted sequence captures what a reasonable response should look like. Some faults exist however–consider the question type comparison_sort. An example: 'Sort -1, 0.3, -6, -24/11, 3, 5, 1 in descending order.' with primary prediction '5, 3, 1, -0.3, -1, -24/11, -6'. Observe the -0.3 in the prediction output–a value which is not even an option given in the problem statement! Such an example would not make a worthwhile distractor as it fails to test for the mathematical notion this question fails to over a high partial credit score. In table 4 we provide a qualitative review per question type of character offset predictions for multiple choice question distractor generation. By comparing to their respective partial credit score we find that generally a high score is an indicator for character offset predictions to also be reasonable.

4.2.2 Character Offset DG: Interpolate Sampling

The following are a couple of curated model responses–the order of the distractors matches the ordering of the probability of the initial character offset. So the first value listed is the model's primary greedily decoded prediction, the second is the sequence generated when we force the initial character to be the second most probable, and so on. If the model predicts the correct solution it is bolded.

'solve -2*s - 40 = -2*j, 53*j - 62*j + 245 = 4*s for s.'
output: **5**, -5, 4, 1

'let i(a) = -a**2 + 1319*a - 22130. calculate i(17).'
output: **4**, -4, 5, 14

Here we see a pair of ideal outputs from the algebra_linear_2d and polynomials_evaluate sets respectively. Not only did the model successfully predict the correct solution sequences of 5 and 4, but also produced noteworthy distractors. Observe the similar magnitude and the prediction of -5 and -4–sign changes of the true solutions and a powerful arithmetic skills check.

Quantitative measurement of the potential for character offsetting in generating multiple choice question distractors is a difficult endeavor due to the lack of formal definition of the problem. However, we observe general groupings of questions which seem to have potential for the use of character offsetting as a computationally cheap method of doing so. Table 4 is a qualitative ranking when given a random subset of outputs for each interpolate question type, whether offset predictions are not reasonable (low, not reported), whether at least one offset sequence is reasonable (medium), or whether most offset sequence predictions are reasonable (high). We view a prediction as unreasonable if it is mathematically meaningless or as mentioned in 4.2.1 either unreasonable given the problem's context or disqualified due to format. We find a higher partial credit score of the primary predictions frequently but not always aligns with whether offset predictions would also be reasonable.

Table 3 shows the comparison of the model generated distractors and rule based distractors for five questions. We generated four distractors for each question from the model and generated ten distractors from the rule based system. In the given table we show the most matching distractors. As we can see we get two matches each for questions (1), (4) and (5). There are three matching distractors for question (3) while there is no matching distractor for question (2). Despite being no matching distractor, our model gets the form of the distractors correct, and the predicted distractors at a glance can be used on a real quiz. In question (4), we can see that the model gets the multiplicity of 120 right and tries to stay around it. From the above examples we can see that our model first tries to get the form of the answer correct and then aims for computational compositionality.

4.2.3 Character Offset DG: Standard Exam Sampling

We sample several actual standardized exam questions from the SAT, ACT, and a college algebra midterm. Questions are altered before being fed to a model so that mathematical syntax matches Python's. Interestingly models appear resilient to significant changes in the question's formulation. Correct exam solutions are bolded, and generated options are in order of the probability of the initial character offsetting. The exam distractors are provided for comparison below but are removed before the question is fed to a model.

'What is the greatest common factor of 42, 126, and 210?
A) 2 B) 6 C) 14 D) 21 **E) 42**'
output: 42, 6, 21, 12

An interesting example [1] as the numbers_gcd data set only ever presents two values to find the gcd of, while the above presents three. Not only does the model predict the correct solution, but two distractors also used in the actual exam.

'Evaluate the function f(x) = 3 + (x-5)**(1/2) at x = 9. A) 1 **B) 5** C) 6 D) 7'
output: 5, 49, -5, 9

Again we observe relatively reasonable responses given question formulations which diverge significantly from the templates trained on. This question [25] is similar to a polynomials_evaluate type. However, the difference is no fractional powers exist in the train-easy set–yet even with the power symbol being replaced by the unknown character the model still generates valid distractors (and the correct solution, but this is clearly by chance as the model has no knowledge of roots).

5. CONCLUSION

5.1 Summary

Two experiments quantitatively showed that a GRU has mixed results when attempting to solve elementary math problems. Our alternative goal of multiple choice distractor generation for several question types typically found in pre-undergraduate education by applying a modified greedy decoding schema referred to as character offsetting was successful. Evaluation using an edit distance based partial credit scoring metric as opposed to a binary one demonstrates greatly increased consistency and performance for capturing a reasonable response. We found the following:

- Generally the easiest math problem types for a GRU is comparison tasks which is not surprisingly since this is a fundamental problem encountered early in education. It would appear the ability for GRUs to abstract mathematical knowledge is minimal.
- The ability for networks to capture the essence of a reasonable response for several question types is shown. Leveraging the proposed practice of character offsetting we show that these networks can cheaply generate distractor options for multiple choice questions.

5.2 Future Work

It would be interesting to compare a beam search decoding's non principle predicted sequences to those produced by character offsetting and whether for certain question types more worthwhile distractors are produced. The general capability for character offsetting to produce at least one worthwhile distractor for the medium potential questions listed in table 4 hint that with some refinements to the decoding schema or training parameters could potentially become high potential question types.

6. ACKNOWLEDGEMENTS

We gratefully acknowledge support from Teaching and Learning with Technology at The Pennsylvania State University.

Supplementary Material - Appendix

S.No.	Question	Answer	Distractors (Model)	Distractors (Rule Based)
1	express $-105c^2 - 5c^3 + 7c - 50c + 41c + 332c^2$ in the form $rc^3 + ic^2 + b + uc$ and give u.	-2	-1 0 3 2	-9 0 -16 2
2	let $k(w) = 2w^2 - 4w^2 + 3w^2$. let $z = -29 + 33$. let $r(o) = -4 - 39o^2 + 84o^2 + z$. give $r(k(s))$.	$45s^4$	$-s^4$ $8s^4$ $9s^4$ $18s^4$	$-45s^4$ $45s^4 - 4$ $135s^4$ 0
3	let u be $1/(114/56 - 2)$. suppose $-3d = -3p + 24$, $-16d - u = -4p - 13d$. solve $c - 3z + 2 = -c$, $0 = -4c - pz - 4$ for c.	-1	1 4 2 0	1 -2 2 0
4	let v be $4/(-14) + -1 * (-596)/28$. let $a = v - 15$. what is the third derivative of $-6b^2 - 9b^6 + 23b^a - 8b^6$ wrt b?	$720b^3$	$-120b^3$ $120b^3$ $60b^3$ $240b^3$	$-120b^3$ $120b^3$ $-720b^3$ $360b^3$
5	let $a(h) = 2h^2 - 9h + 4$. suppose $-26w - 20w = -55w + 126$. what is the remainder when $a(-7)$ is divided by w?	11	18 8 21 9	7 8 3 9

Table 3: Qualitative Comparison of Distractors Generated using Neural Network Model and Rule Based System

Potential	Question	Encoder Hidden Size	Batch Size	Partial Credit Score	Mean Score
High	algebra_linear_2d_composed	2048	128	0.837	0.766
	algebra_linear_2d	2048	256	0.811	
	algebra_linear_1d_composed	2048	128	0.887	
	algebra_linear_1d	2048	256	0.694	
	algebra_sequence_next_term	2048	128	0.774	
	arithmetic_mul_div_multiple	2048	256	0.772	
	arithmetic_nearest_integer_root	2048	256	0.730	
	polynomials_evaluate_composed	2048	128	0.754	
	polynomials_evaluate	2048	128	0.709	
	polynomials_expand	512	128	0.642	
	polynomials_coefficient_named	2048	128	0.733	
	numbers_gcd_composed	2048	128	0.760	
	numbers_gcd	2048	256	0.762	
	numbers_lcm_composed	2048	128	0.737	
	numbers_div_remainder_composed	2048	128	0.800	
	numbers_div_remainder	2048	256	0.762	
	numbers_place_value_composed	2048	128	0.859	
Medium	algebra_sequence_nth_term	1024	128	0.507	0.624
	arithmetic_add_or_sub	2048	256	0.501	
	arithmetic_mul	2048	256	0.451	
	arithmetic_div	2048	256	0.559	
	arithmetic_mixed	2048	256	0.688	
	arithmetic_add_sub_multiple	2048	256	0.755	
	arithmetic_add_or_sub_in_base	2048	256	0.682	
	calculus_differentiate_composed	1024	128	0.589	
	calculus_differentiate	512	128	0.730	
	measurement_time	2048	256	0.838	
	numbers_lcm	2048	256	0.564	

Table 4: Qualitative ranking of the potential for models to use character offsetting for generating distractors based on observed predictions on interpolate. Questions not listed are those whose predictions were generally unreasonable as defined in 4.2.2 or disqualified due to formatting mentioned in 4.2.1. Model specifications are included as well. Decoder hidden size was 2048 for all models. Encoder/Decoder embedding dimension and number of attentional heads was 512/512 and 16 respectively.

7. REFERENCES

[1] ACT. Practice test questions: Math, 2020.

[2] I. Aldabe and M. Maritxalar. Automatic distractor generation for domain specific texts. In *International Conference on Natural Language Processing*, pages 27–38. Springer, 2010.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[4] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[5] J. A. Erickson. Natural language processing for open ended questions in mathematics within intelligent tutoring systems. In *EDM*, 2020.

[6] M. Ganesalingam. The language of mathematics. In *The language of mathematics*, pages 17–38. Springer, 2013.

[7] H. C. Goodrich. Distractor efficiency in foreign language testing. *TESOL Quarterly*, 11(1):69–78, 1977.

[8] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Four volumes. Springer, 2014.

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[10] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

[11] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[12] G. Lample and F. Charton. Deep learning for symbolic mathematics. *CoRR*, abs/1912.01412, 2019.

[13] C. Liang, X. Yang, N. Dave, D. Wham, B. Pursel, and C. L. Giles. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 284–290, 2018.

[14] C. Liang, X. Yang, D. Wham, B. Pursel, R. Passonneaur, and C. L. Giles. Distractor generation with generative adversarial nets for automatically creating fill-in-the-blank questions. In *Proceedings of the Knowledge Capture Conference*, page 33. ACM, 2017.

[15] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

[16] T. Liu, Q. Fang, W. Ding, Z. Wu, and Z. Liu. Mathematical word problem generation from commonsense knowledge graph and equations. *ArXiv*, abs/2010.06196, 2020.

[17] J. J. Michalenko, A. S. Lan, and R. G. Baraniuk. Personalized feedback for open-response mathematical questions using long short-term memory networks. In X. Hu, T. Barnes, A. Hershkovitz, and L. Paquette, editors, *Proceedings of the 10th International Conference on Educational Data Mining, EDM 2017, Wuhan, Hubei, China, June 25-28, 2017*. International Educational Data Mining Society (IEDMS), 2017.

[18] V.-M. Pho, A.-L. Ligozat, and B. Grau. Distractor quality evaluation in multiple choice questions. In C. Conati, N. Heffernan, A. Mitrovic, and M. F. Verdejo, editors, *Artificial Intelligence in Education*, pages 377–386, Cham, 2015. Springer International Publishing.

[19] O. Polozov, E. O'Rourke, A. M. Smith, L. Zettlemoyer, S. Gulwani, and Z. Popović. Personalized mathematical word problem generation. In *IJCAI 2015*, May 2015.

[20] S. Roy and D. Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics.

[21] S. Roy and D. Roth. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3082–3088. AAAI Press, 2017.

[22] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli. Analysing mathematical reasoning abilities of neural models. *CoRR*, abs/1904.01557, 2019.

[23] Y. Susanti, T. Tokunaga, H. Nishikawa, and H. Obari. Automatic distractor generation for multiple-choice english vocabulary questions. *Research and Practice in Technology Enhanced Learning*, 13(1):15, 2018.

[24] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[25] University Department of Mathematics. Math 021 sample exams, exam 1 sample exam a, 2020.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[27] L. Wang, D. Zhang, L. Gao, J. Song, L. Guo, and H. T. Shen. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[28] L. Wang, D. Zhang, J. Zhang, X. Xu, L. Gao, B. T. Dai, and H. T. Shen. Template-based math word problem solvers with recursive neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7144–7151, Jul. 2019.

[29] Y. Wang, X. Liu, and S. Shi. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[30] K. Zaporojets, G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. Solving math word problems by

scoring equations with recursive neural networks. *arXiv preprint arXiv:2009.05639*, 2020.

Linguistic and Gestural Coordination: Do Learners Converge in Collaborative Dialogue?

Arabella J. Sinclair
ILLC, University of Amsterdam
Science Park 107
1098 XG Amsterdam
a.j.sinclair@uva.nl

Bertrand Schneider
Graduate School of Education, Harvard
13 Appian Way
Cambridge, MA 02138
bertrand_schneider@gse.harvard.edu

ABSTRACT

Collaborative dialogue is rich in conscious and subconscious coordination behaviours between participants. This work explores collaborative learner dialogue through theories of alignment, analysing inter-partner movement and language use with respect to our hypotheses: that they interrelate, and that they form predictors of collaboration quality and learning. In keeping with theories of alignment, we find that linguistic alignment and gestural synchrony both correlate significantly with one another in dialogue. We also find strong individual correlations of these metrics with collaboration quality. We find that linguistic and gestural alignment also correlate with learning. Through regression analysis, we find that although interconnected, these measures in combination are significant predictors of collaborative problem solving success. We contribute additional evidence to support the theory that alignment takes place across multiple levels of communication, and provide a methodological approach for analysing inter-speaker dynamics in a multimodal task based setting. Our work has implications for the teaching community, our measures can help identify poorly performing groups, lending itself to informing the design of real time intervention strategies or formative assessment for collaborative learning.

Keywords

Dialogue, Gesture, Natural Language Processing, Alignment, Collaborative Learning

1. INTRODUCTION

Collaborative problem solving has long been a focus of educational research, and has been deemed an educational learning objective of critical importance in the 21st century workforce [12]. In the education literature, collaboration success is often analysed with respect to a *joint problem space* as created through learner interaction [31]. This joint problem space integrates learner shared goals, descriptions of the problem state, awareness of available problem solving actions and the associations between these aspects. The emergence of this shared conceptual space is constructed through shared language, situation and activity.

Alignment in dialogue, the language component of this interaction, is also commonly attributed to an automatic mechanism to achieve a shared understanding, or *situation model* [27]. This account of dialogue predicts alignment across various modes of communication, from word level to gesture and gaze patterns. Specifically in a task based setting, alignment is thought to aid mutual understanding [10, 27]. These theories of alignment and collaborative learning when taken together suggest that convergence at many levels of communication will take place in parallel over the course of collaborative dialogue, and that this alignment will be indicative of collaborative success. Additionally, the collaborative learning literature suggests that the effort necessary to build shared understanding is what actually leads to learning [40], thus alignment, already found to be predictive of student learning in a teacher student context [42], may also be indicative of this.

Investigating collaborative problem solving through the lens of alignment can give additional insights to this complex problem of convergence [38]. In this work, we examine the synchrony and convervence between students at both a linguisitic and gestural level, via inter-student metrics of linguistic alignment and movement synchrony. Of particular interest in this study is the separate coding of collaboration and learning in the learner dialogues. This allows for side by side comparison of the different modalities, and the analysis of their interaction with respect to these outcomes. Gestural and linguistic coordination between locutors has long been linked in dialogue, both properties having been individually explored for facilitating collaboration and learning in various settings.

We offer an exploration of theoretically motivated metrics to capture synchronisation and alignment at the levels of linguistic expressions and movement patterns. We explore correlations between the measures themselves and between collaboration and learning. Exploring the relationship between these measures we find strong correlations between modalities, in line with the collaboration and alignment literature. Finally we explore the combination of these modalities in their predictive power for both collaboration and learning, finding that although they are interrelated, each plays a significant role in prediction quality.

Arabella Sinclair and Bertrand Schneider "Linguistic and Gestural Coordination: Do Learners Converge in Collaborative Dialogue?". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 431-438. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

1.1 Research Questions

Motivated by the hypotheses we draw from the literature on collaboration, learning and alignment, we hypothesise that more successful learner dyads will converge in both their language use and their movement patterns to a visible degree, as the students align their mental models during the learning process. We also hypothesise that together, these aspects of interaction can provide useful tools in the analysis of student learning. We split our analysis into the following research questions:

RQ1: ***Evidence of Convergence:*** Are linguistic alignment, gestural synchrony and convervence effects between students higher than by chance task and vocabulary effects?

RQ2: ***Convergence Vs Collaboration & Learning:*** How do our measures of convergence correlate with collaboration and learning?

RQ3: ***Interaction: Linguistic Vs Gestural:*** Do these modalities correlate with one another?

RQ4: ***Multimodal: Do combined measures of synchrony predict learning and collaboration outcomes:*** are the measures in combination predictive of learning and collaboration?

2. BACKGROUND

Evidence of convergence. Experimentally, in a two person dialogue setting, speakers have been found to converge, each aligning to their locutor at many levels of communication: lexical, structural, gesture and conceptual[27]. Speakers have been found to spontaneously coordinate body postures and gaze patterns during conversation [35]. Behaviour matching in multimodal communication has also been found to be temporally synchronised in collaborative task-based activity, when participants are facing each other [22]. Imitation or mimicry between people unconscious of this behaviour has been found in incidental mannerisms such as the bouncing of a foot, or rubbing a nose [9]. People imitate one another in dialogue across many different modalities, including lexical choice [17], accent [18], pauses [7], speech rate [43] and syntax [30, 4, 26]. This imitation has been linked to having social benefits, for example, [9] find that speakers in those pairs where their incidental mannerisms were mimicked perceived the interaction as running more smoothly than those whose were not. In terms of collaboration, multi modal behaviour matching has been found to occur in a synchronous manner in a task based collaborative dialogue where rapport and its role in learning and convergence was investigated [22]. Acoustic prosodic entrainment has also been found to correlate with rapport, a social quality of the interaction, in collaborative learning dialogues [23]. Parallel to this, it has been argued that infants' early skills of joint attention is their emerging understanding that other people exist as intentional agents [8], as they develop theory of mind. In terms of learning, lexical entrainment has been shown to correlate with success in multiparty student engineering group project meetings [16], where higher scoring teams were more likely to increase their entrainment in project words over the course of a dialogue, while lower scoring teams are more likely to diverge. Alignment level has been shown to vary with student ability [36], and convergence of lexical and speech features from student to tutor in spoken tutorial dialogue corpora has been shown to be a useful predictor of learning [42].

Language and gesture in learning. Gesture has an important role in teaching and learning [32], as does language, which, at a structural level, has been shown to exhibit effects characteristic of both learning and implicitness, thought of as an aspect of alignment or coordination between interlocutors [15]. A wide range of lingusitic features derived from student dialogues have been found to be effective predictors of both learning gains and collaboration quality [29]. Categorising gesture in an educational setting often adopts the framework proposed by [24], of separating them into four basic types: beat (gestures devoid of topical content yet which lend temporal or emphatic structure i.e. hand tapping, head movement for emphasis), deictic (concrete or abstract pointing i.e. to match an object referred to as 'this' or 'that', or a concept in the past that is being referred to), iconic (also referred to as representational, i.e. making a gesture of putting a phone to ones ear), and metaphor (gestures to illustrate abstract concepts, such as moving hands together to illustrate mathematical convergence, or drawing a trend line in the air to demonstrate positive correlation). While gestures are pervasive, not all types are equally represented in particular speech events, and are very dependent on the dialogue context, however, various studies have found that gesture and speech together provide a better index of mental representation than speech alone, and to be an important aspect in learning [19, 11].

3. METHODS

Speaker	*Utterance*
Left:	then **we need** to **turn left** . again put an if **and then** turn the view book .
Right:	so another if do statement ?
Right:	was it just when you write in **the sensor** here , right into that . i say talk to motor ? all right , a and b .
Left:	if it is that , then we take a left . then **turn left** .
Right:	**turn left** , which is right here , right , so
Left:	put this inside that **and then** again , **we need** to turn left .
Right:	another if statement ? remember , control . **and then** talk to motor again . turn left
Left:	**turn left** comes after .
Right:	and **we need** one for ...

Table 1: Example dialogue excerpt. Expressions in bold indicate shared **lexical** constructions.

Experimental Setup. The experimental setup consisted of 40 pairs of undergraduate students participating in a collaborative problem solving task of programming a robot to traverse a maze. The participants had no prior programming experience. The participants sat facing a computer screen, and were recorded as they worked through the shared exercises. During the collaborative aspect of the task, the focus of our analysis in this work, participant dialogue was recorded and subsequently transcribed. Body movement data was also recorded via a Microsoft kinect sensor. This resulted in timestamped language and movement data for the 30 minute period of the task. The participants were individually given a pre and post test on a similar set of exercises, in order to evaluate the relative learning in the dyads. The learning assessment consisted of four short answer or fill-in-the-blank questions that assessed their understanding of basic computer science competencies (adapted from [5, 44]). Learning gains were computed by subtracting pre-test scores from post-test scores and divided by the total number of points to be gained minus the pre-test [13]. Collaboration was evaluated on a series of axes derived from the

collaboration assessment measures proposed in [25]: *sustaining mutual understanding, dialogue management, information pooling, reaching consensus, task division, time management, technical coordination, reciprocal interaction, and individual task orientation*[1]. Each dimension was given a score between +2 and -2 (each score was defined with concrete behaviors in a codebook). Researchers double-coded 20% of the sessions and had a Cronbach's alpha of .65 (75% agreement). An overall measure of collaboration was defined as the aggregate of all collaborative features. More details of the study, the data, experimental setup and coding of collaboration and learning scores can be found in Reilly et al. 2019 [29].

Dialogue transcripts. An example snippet from a dialogue with high collaboration score is provided in Table 1. The transcripts occasionally include some utterances from the facilitator, for whom we are not interested in measuring any alignment effects. The facilitator typically will speak most at the beginning of the dialogue, thus we remove all intial utterances (including participant) which interleave facilitator and participant. For the remainder of the dialogue, facilitator utterances are simply removed. Punctuation, although added at the annotators discretion, is retained, as it provides valuable information about the pace of the language used, and indicates the fragmented nature of these dialogues. The transcripts were tokenised before analysis using the nltk python package[2].

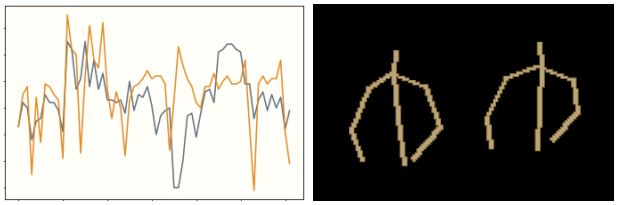


Figure 1: Example student movement pattern over time (left) and student geometries (right).

Movement Data. Student movement was processed using motion sensors from microsoft kinect, returning a series of geometries representing the students' body positions in space. An example of two students sitting at the table (which obsures the lower half of their bodies) can be seen in Figure 1. We only include data from the time period where the students were performing the collaborative activity, which corresponds to the transcript data of 30 minutes per session. In terms of pre-processing choices, our interest in the movement data is where the students mimic the gestures of their partner independent on their position relative to the camera or one another. This allows us to abstract from the postural shapes, relative size, and dominant hand of the participant's gestural patterns. We thus use averaged 30-second time slices[3] of the movement data (a measure of between frame positional difference). We further process this data to account for the differences in movement which the experimental setup introduces: we apply standardisation[4] to each participant signal in order that two signals of different means and standard deviations can be compared on the same axis. This grants us a measure of variance similarity, which captures better the elements of beat gesture patterns separate from absolute movement differences, as we know the students consistently display different mean movement levels across dyads.

3.1 Computing Lexical Alignment

We operationalise linguistic alignment in this work at the lexical (word) level, derived from the dialogue transcripts, extracting *shared expressions*, which we define as any sequence of tokens which contain at least one word (e.g. single punctuation marks are excluded). The automatic extraction of shared expressions per dialogue is an instance of the longest common sub-sequence problem [20, 3]. For each dialogue, we extract the inventories of shared expressions using the method proposed by Duplessis et al. [14]. For each of the two dialogue-specific inventories of shared constructions, we compute the following measures:

- *Expression Variety (EV):* The lexical diversity of the expression vocabulary.
- *Expression Repetition (ER):* The ratio of produced tokens belonging to an instance of an established expression
- *Vocabulary overlap (VO):* Captures the richness of the shared vocabulary, the ratio of shared vocabulary present in the dialogue between participants:

$$\frac{\#(words_{\text{speaker1}} \cap words_{\text{speaker2}})}{\#(words_{\text{speaker1}} \cup words_{\text{speaker2}})}$$

Individuals repeat and introduce expressions at different rates within dyads, thus we additionally calculate dyad level measures to capture the symmetry between interlocutors.

- *Expression Initiator (IE) Difference:* Difference in % of shared constructions introduced by each dialogue participant. *Initiator* describes the dialogue participant to first use a subsequently shared and repeated construction.

  $||IE(speaker\ one) - IE(speaker\ two)||$
- *Expression Repetition Difference:* The difference in proportion of an individual speaker's utterances which contain a usage of a shared construction:

  $||ER(speaker\ one) - ER(speaker\ two)||$

These measures capture the between speaker repetition within dialogue, which we use as a proxy for measuring the coordination or alignment between the speakers. An example of expression repetition can be found in Table 1

[1]Detailed descriptions of these measures used can be found in [25]

[2]NLTK[21] python package http://www.nltk.org

[3]Our choice of 30-second slices was in part informed by previous work [38], and through qualitatively examining the data at 5-second and 30-second slices, 30-seconds seems to be sufficient to capture interesting aspects of finer-grained hand movement, but not so fine as to render average total movement data meaningless

[4]Standardising a time series dataset involves re-scaling the distribution of values, also known as *Z-normalisation*

3.2 Computing Gestural Synchrony

We use Dynamic Time Warping [33] as the measure of similarity between partner movement patterns as it has been found to be a consistently robust measure of time series similarity [2], which although introduced as a method for analysing speech signal [33], has been employed successfully in the field of gesture recognition [1]. DTW is a technique to find the optimal alignment between two time series, through the stretching or compression of either series along its time axis. This warping can be used to find corresponding regions between two time series, and serve as a distance measure. This measure of distance allows us to capture slightly out of sync movement patterns, or those of slightly differnt phase length, to be deemed more similar than their difference in slope would suggest. Our main motivation for choosing DTW over other metrics demonstrated suitable for this task such as [38, 28] is our hope to capture the similarity of slightly asynchronus movement, which, if indeed movement and linguistic alignment is linked, should follow a similar mixed turn taking behaviour as dialogue [41]. As well as providing a robust distance measure between two time series, DTW returns a warping path which describes in what direction the time series needs to be moved in order to best align: in other words, the warping path can provide useful information about leader and follower dynamics which we exploit in our analysis.

Inspired by common measures of gestural synchrony/body language, we investigate the following dyadic behavioural properties:

- *Movement Difference (Mdiff):* Global mean movement difference in a pair. Calculated as the absolute value of the difference between the means of each participants.
- *Movement Synchrony (dtw_dist):* The synchrony between pairs in terms of their movement patterns, as measured by the Dynamic Time Warping (DTW)[33] distance
- *Leader Follower dynamics (diffLF):* The directionality of the alignment of the movement between the pair. This metric is derived from the DTW path.

Additionally, to measure whether these similarities become more pronounced as the session progresses, we divide the session in half by timestamp, and compute the measures per half. We use the difference between dialogue halves (*second - first*) as a measure to capture convergence. This results in three additional measures corresponding for those synchrony measures above: *Mdiff_change*, *dtw_dist_change* and *diffLF_change*. The aspects of the body geometries which we focus on consist of the points for *Head, Hands, Shoulders* and *Total* (average) movement. For hand and shoulder measures, these are defined as an average of the movement in the right and left points for each aspect.

3.3 Measure Validation - Baseline

A certain level of similarity between speakers will exist independently of their adapting to one another. Due to their performing the same task, vovabulary will necessarily be constrained by topic, and consistent across pairings. Due to the experiment configuration, task specific gesture patterns such as moving the robot, or interacting with the computer, as well as to which side of them their interlocutor is will also lead to movement similarities, e.g. turning to the right vs the left to speak. We thus create baselines for both dialogue and movement data which demonstrate the levels of similarity inherent to the task setup. For the dialogue baseline, we create a scrambled version of the corpus by retaining the utterances of one of the students and interleaving it with utterances randomly drawn from another pair, per speaker. For a partner specific movement baseline, the movement data from each student is randomly paired with the data from another student on the same side relative to them as their partner was (i.e for each participant on the right hand side, replace their partner with a participant from the left hand side). To further check task specific effects of the seating configuration, we pair students sitting in the same position with one another, in order to confirm that the role does not show more similarities than the origional student pairings.

4. RESULTS

4.1 Analysis 1: Measuring Convergence

Linguistic. We firstly hypothesise that there will be significant inter dyad repetition beyond what the task demands by chance, since alignment has been linked to both learning, as well as collaboration, and this same measure has found significant alignment levels in negotiation[14], as well as in second language tutoring dialogue[37], although this dialogue setting is different since both speakers are learners. Firstly we explore whether alignment is greater than by chance: we therefore compare the original dialogues to the shuffled baseline in the same manner as [14]. The expression variety is significantly higher for the original (mean=0.118, std=0.023) than for the shuffled dialogues (mean=0.110, std=0.015). Statistical difference is checked by a Wilcoxon rank sum test (U = 1141, $p = 0.03 < 0.05$, r = 0.21)[5] This indicates that there exists a richer and more dyad specific expression lexicon. The expression repetition is also significantly higher for the original (mean=0.509, std=0.123) than for the shuffled dialogues (mean=0.487, std=0.109) (U = 1079.5, $p = 0.014 < 0.05$, r = 0.25). This means that the level of repetition between student dyads is not simply incidental, and can be attributed to alignment or routinisation effects. Finally, as a measure of how task specific the vocabulary is, we find the vocabulary overlap between speakers significantly higher in the original (mean=0.509, std=0.123) than in the shuffled dialogues (mean=0.487, std=0.109) (U = 856, $p = 0.0002 < 0.001$, r = 0.41). This difference demonstrates that students share a much richer vocabulary than would happen by chance in performing this task. Overall, these results show that the collaborative student dialogues constitute a richer expression lexicon than they would by chance, indicating that the students align to one another, resulting in their langauge converging [10, 27].

Gestural. We hypothesise that our measures of movement matching will result in higher partner-specific synchrony than

[5]Following [14], for each test, we report the test statistics (U/W), the p - value (p) and the effect size (r)

in our baseline: i.e. lower distance between pairs collaborating than simply any student performing the same task with a different partner. We find that within dyad DTW similarity is significantly higher than in both the partner substitution baseline (t= -2.0401, $p < 0.05$), and the within side baseline (t = -2.0397, $p < 0.05$). This indicates DTW is a useful measure of movement similarity in this setting showing that this method is suitable for capturing partner specific effects of these movement patterns and can allow us to use this to compare similarities between dyads.

4.2 Analysis 2: Convergence Correlated with Collaboration and Learning

Lingusitic alignment. We hypothesise that alignment between students will correlate with learning, since in other tutoring settings, it has been found to correlate with both learning gains [42] and linguistic ability [36]. Additionally, global language features of collaborative dialogue have also found to correlate with learning gains [29]. To answer *RQ2*, we compare our linguistic alignment metrics with learning and collaboration scores. We find support for our hypothesis about collaboration and alignment correlating. We find ER (r 0.680 $p < 0.001$), EV (r 0.622 $p < 0.001$), and VO (r 0.663, p <0.001) all correlate with collaboration using Pearson's r correlation coefficient. This shows that inter partner repetition is important, and that students will converge even to less common language in a collaborative setting. We also find our between learner measures significantly correlated with collaboration, IE_diff (r=-0.570, $p < 0.01$) and ER_diff (r=-0.54, $p < 0.001$) meaning that smaller differences between learner initiation and repetition of shared expressions correlates with how well they collaborate. EV (r=0.442, p =0.026), ER (r=0.442, p =0.006) and VO (r=0.349, p =0.034) also correlate with learning, although to a lesser degree. An intuition as for why, is that in other studies reporting alignment correlation with learning analyse dialogues conducted in an asymmetric *tutoring* setting, where adopting the language of the teacher is a sensible learning strategy as it is assumed that this language is correct. In our case, since these dialogues are between *peer* learners, the learning outcome is somewhat dependent on the rapport within the dyad, and the information aligned to being correct. In other words, in some cases, the learners may be converging to a shared mental representation, but it may not be the correct one. In keeping with this observation of dyad rapport and equality, IE_diff (r=-0.487, $p = 0.002$) and ER_diff (r=-0.515, $p = 0.001$) both show strongly that more equal contributions from the students in terms of repeating one another, and in introducing words upon which to align correlate with learning.

Movement Synchrony and Convergence. We hypothesise that movement synchrony and convergence, as defined by DTW distance and its change over the interaction, will provide a robust measure of synchrony which will better distinguish between dyads with differing activity levels, which in turn should correlate with collaboration and learning, in keeping with previous results with other measures in task based dialogue [22, 32, 28]. We compare our movement similarity metric with learning and collaboration scores. Overall

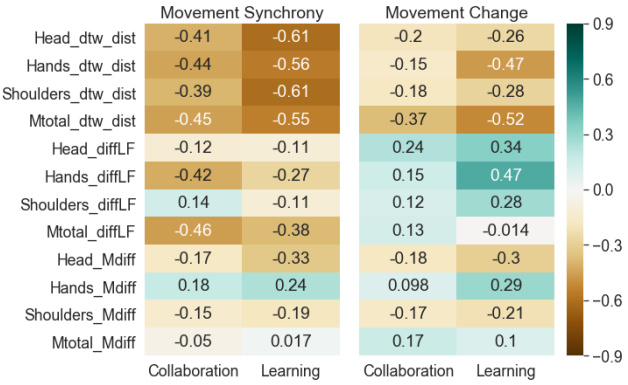


Figure 2: Gestural synchrony and convergence vs. Collaboration and Learning measures correlation with Pearson's r. Significant p values reported in the text.

as can be seen from Figure 2, we find average movement synchrony to correlate with both collaboration and learning. In terms of learning, DTW_dist measures for Head (r = -0.611, $p > 0.001$) Hands (r = -0.561, p = 0.002), Shoulders (r = -0.609, $p > 0.001$), and Mtotal (r = -0.611, p =0.006) all significantly correlate with learning to a strong degree. Convergence between dyads in terms of Mtotal (r=-0.519, p = 0.006) and Hands(r=-0.467, p = 0.014) also significantly correlate with learning. Finally, dyads becoming more dissimilar in terms of hand movement (having a stronger leader follower dynamic) also significantly correlates with learning: Hands_diffLF (r=0.471, p = 0.013). With collaboration, Head Hands Shoulders and Mtotal all significantly ($p < 0.05$) correlate. As does the diffLF for Hands (r=-0.42, p = 0.029) and Mtotal (r=-0.519, p = 0.006). Overall, the results are intuitive: we find that more synchronus pairs as measured by DTW distance significantly correlate with collaboration quality. We also find that convergence between dyads is present (negative correlation between *dtw_dist change* and *Mdiff change* show greater similarity between learners over time) and correlates with learning quite strongly for some movement metrics. We also see a positive correlation between the *diffLF change* features, particularly with learning, indicating that while convergence of behaviour is important, some aspects of turn taking and initiative are separate to this.

4.3 Analysis 3: Comparing Linguistic and Gestural Convergence

Comparing Linguistic and Gestural convergence, we hypothesise these aspects of communication will correlate with one another, as previous literature suggests [17, 4, 22]. To answer research question (*RQ3*), we contrast the modalities themselves. We split this comparison to compare gestural and linguistic coordination. We hypothesise that movement synchrony and linguistic alignment will correlate strongly, due to the process of speakers' alignment of shared mental representations taking place across various linguistic and paralinguistic levels [6, 27]: if dyads align at the lexical level, it is likely that the same process leading to this alignment will affect the gestural level also [27]. The DTW path allows us to capture the relationship between slightly offset movement patterns of beat gesture mimicry [24], and the

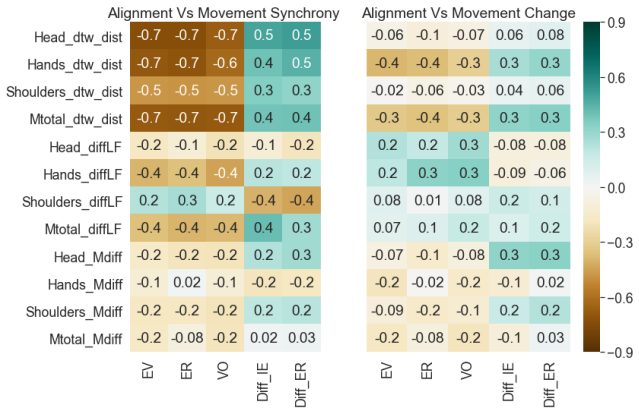


Figure 3: Movement measures vs. linguistic alignment measures. Pearson's r correlation coefficient.

case where linguistic alignment in turn taking utterances is low, which can lead to more synchronised patterns of movement [32]. Previous work has also found evidence of the coordination of of lexical alignment and gestural behaviour in a multimodal [35, 9, 22] context, we thus hypothesise that collaborative problem solving dialogues will also demonstrate this. We find significant correlation between linguistic alignment measures and movement synchrony across all movement patterns (Figure 3), with strongest effects for head (r = -0.735, p-value: 1.225) and hands (r = -0.706 p-value: 3.811), providing support that our hypothesis about lexical patterns influencing gestural alignment at the level of beat gesture and head nodding may be be true for this setting. In terms of convergence, there is a significant correlation with change in hand movement and expression variety (r = -0.387, p-value: 0.0458). Interestingly, the difference between speakers lingusitic (*Diff_IE*, *Diff_ER*) patterns positively correlates with both their difference in movement synchrony and speaeker divercence, indicating that asymmetric relationships between students are visible across modalities of communication. Inkeeping with our hypothesis we find strong negative correlation between between speaker difference in movement and divergence (strong correlation between similarity and convergence) with the linguistic measures of convergence, providing supporting evidence for the hypothesis that lingusitic and gestural convergence are part of the same underlying communicative process.

4.4 Analysis 4: Predicting Learning and Collaboration

Finally, in order to answer research question (*RQ4*), to find combined interaction effects of the various inter modality measures, we fit a series of mixed effect regression models[6]. We hypothesise that while each measure individually is strong, and although the measures themselves are correlated, each modality will provide its own distinct information contributing to learning and collaboration aspects. We perform backward step wise model selection to select the best predictors, firstly fitting each model with all relevant variables and stopping only when all remaining terms have significance $p < 0.05$. Although RMSE and r^2 values of predicted data are highest when combining all factors, we wished to discover the minimally significant descriptive set of criteria in order to find more interaction in our results.

[6]To fit the data and perform the statistical tests within this paper, we use the Statsmodels python package [34]

Table 2: Mixed effects Regression model multimodal results

Formula
Learning: **RMSE:5.48** r^2**:0.90**
*Learning ~ EV : ER + Diff_IE * Diff_IE + Handmean_movement_30_diffLFChange + Shouldermean_movement_30_diffLFChange*
Collaboration **RMSE:0.15** r^2**:0.999**
*Collaboration ~ EV * ER + Diff_IE * Diff_ER + Head_movement_30_dtw_dist + Handmean_movement_30_dtw_dist + Shouldermean_movement_30_dtw_dist + movement_total_30_dtw_dist + Head_movement_30_diffLFChange + Shouldermean_movement_30_diffLFChange + movement_total_30_diffLFChange + Head_movement_30_dtw_dist_change + Handmean_movement_30_dtw_dist_change + Shouldermean_movement_30_dtw_dist_change + movement_total_30_dtw_dist_change + Head_movement_30_diffLF + Handmean_movement_30_diffLF + Shouldermean_movement_30_diffLF + movement_total_30_diffLF*

Table 2 shows the minimal significant set of linguistic and gestural factors and their interaction in terms of their ability to predict the dependent variables of *learning* and *collaboration*. Each modality separately can form a good predictor of both alignment and learning in this setting. However, this analysis offers strong support for the multimodal modelling of collaborative problem solving, proving that although correlating with one another, both linguistic and gestural aspects have an independent role to play when predicting learning and collaboration. Broadly, from Table 2, the gestural features chosen indicate that both the measures of synchrony, and those for convergence (*_change* features) play a role in prediction. It is also clear that predicting collaboration in this case is easier than learning. This may be influenced by ceiling effects or ease of pre-test being a limiting factor. e.g. a learner with very good pretest score will have hit a ceiling by the end of the session.

5. DISCUSSION & CONCLUSION

We find significant levels of both linguistic and movement synchrony in our data (*RQ1*). In answer to *RQ2*, we find our measures of linguistic and gestural alignment correlate with collaboration. In terms of learning, we find that the difference in repetition between students negatively correlates with learning, that movement synchrony in general shows strong correlation with learning. In terms of *RQ3*, we find significant strong to medium effects when correlating measures of ER and dtw_dist with one another. This contributes to a growing body of evidence in support of theories of interactive alignment emerging across communicative modalities. Finally, via regression analysis combining our metrics (*RQ4*), we find that although separately powerful, a combination of modalities can best explain collaboration and learning outcomes. Our findings show the importance of analysing between speaker dynamics to capture nuances of learning. Our findings also suggest the use of a multimodal approach for the best understanding of these interactions. We also contribute interesting new evidence adding to work exploring the relationship between linguistic alignment and gestural and movement similarity. Our findings, while limited to a small specific setting, contribute evidence to support existing theories of human cognition and alignment.

6. ACKNOWLEDGMENTS

First author funded under grant agreement No. 819455 from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme.

7. REFERENCES

[1] A. Akl and S. Valaee. Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2270–2273. IEEE, 2010.

[2] A. J. Bagnall, A. Bostrom, J. Large, and J. Lines. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version. *CoRR*, abs/1602.01711, 2016.

[3] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, pages 39–48. IEEE, 2000.

[4] H. P. Branigan, M. J. Pickering, and A. A. Cleland. Syntactic co-ordination in dialogue. *Cognition*, 75(2):B13–B25, 2000.

[5] K. Brennan and M. Resnick. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, volume 1, page 25, 2012.

[6] S. Brennan and H. Clark. Conceptual Pacts and Lexical Choice in Conversation. *Journal of Experimental Psychology*, 22(6):1482–1493, 1996.

[7] J. N. Cappella and S. Planalp. Talk and silence sequences in informal conversations iii: Interspeaker influence. *Human Communication Research*, 7(2):117–132, 1981.

[8] M. Carpenter, K. Nagell, M. Tomasello, G. Butterworth, and C. Moore. Social cognition, joint attention, and communicative competence from 9 to 15 months of age. *Monographs of the society for research in child development*, pages i–174, 1998.

[9] T. L. Chartrand and J. A. Bargh. The chameleon effect: the perception–behavior link and social interaction. *Journal of personality and social psychology*, 76(6):893, 1999.

[10] H. Clark and D. Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22:1–39, 1986.

[11] S. W. Cook, Z. Mitchell, and S. Goldin-Meadow. Gesturing makes learning last. *Cognition*, 106(2):1047–1058, 2008.

[12] N. R. Council et al. *Education for life and work: Developing transferable knowledge and skills in the 21st century*. National Academies Press, 2012.

[13] S. Cuendet, E. Bumbacher, and P. Dillenbourg. Tangible vs. virtual representations: when tangibles benefit the training of spatial skills. In *Proceedings of the 7th Nordic conference on human-computer interaction: Making sense through design*, pages 99–108, 2012.

[14] G. D. Duplessis, C. Clavel, and F. Landragin. Automatic measures to characterise verbal alignment in human-agent interaction. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 71–81, 2017.

[15] V. S. Ferreira and K. Bock. The functions of structural priming. *Language and Cognitive Processes*, 21(7-8):1011–1029, 2006. PMID: 17710210.

[16] H. Friedberg, D. Litman, and S. B. Paletz. Lexical entrainment and success in student engineering groups. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 404–409. IEEE, 2012.

[17] S. Garrod and A. Anderson. Saying what you mean in dialogue: A study in conceptual and semantic co-ordination. *Cognition*, 27(2):181–218, 1987.

[18] H. Giles, P. Powesland, E. A. of Experimental Social Psychology Staff, and E. A. of Experimental Social Psychology. *Speech Style and Social Evaluation*. European monographs in social psychology. European Association of Experimental Social Psychology by Academic Press, 1975.

[19] S. Goldin-Meadow. Beyond words: The importance of gesture to researchers and learners. *Child development*, 71(1):231–239, 2000.

[20] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, 24(4):664–675, 1977.

[21] E. Loper and S. Bird. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[22] M. M. Louwerse, R. Dale, E. G. Bard, and P. Jeuniaux. Behavior matching in multimodal communication is synchronized. *Cognitive science*, 36(8):1404–1426, 2012.

[23] N. Lubold and H. Pon-Barry. Acoustic-prosodic entrainment and rapport in collaborative learning dialogues. In *Proceedings of the 2014 ACM workshop on Multimodal Learning Analytics Workshop and Grand Challenge*, pages 5–12, 2014.

[24] D. McNeill. *Hand and mind: What gestures reveal about thought*. University of Chicago press, 1992.

[25] A. Meier, H. Spada, and N. Rummel. A rating scheme for assessing the quality of computer-supported collaboration processes. *International Journal of Computer-Supported Collaborative Learning*, 2(1):63–86, 2007.

[26] M. J. Pickering and V. S. Ferreira. Structural priming: A critical review. *Psychological bulletin*, 134(3):427, 2008.

[27] M. J. Pickering and S. Garrod. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27(02):169–190, 2004.

[28] J. M. Reilly, M. Ravenell, and B. Schneider. Exploring collaboration using motion sensors and multi-modal learning analytics. *International Educational Data Mining Society*, 2018.

[29] J. M. Reilly and B. Schneider. Predicting the quality of collaborative problem solving through linguistic analysis of discourse. *International Educational Data Mining Society*, 2019.

[30] D. Reitter, F. Keller, and J. D. Moore. Computational modelling of structural priming in dialogue. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 121–124, Stroudsburg, PA, USA, 2006. Association for

Computational Linguistics.

[31] J. Roschelle and S. D. Teasley. The construction of shared knowledge in collaborative problem solving. In *Computer supported collaborative learning*, pages 69–97. Springer, 1995.

[32] W.-M. Roth. Gestures: Their role in teaching and learning. *Review of educational research*, 71(3):365–392, 2001.

[33] H. Sakoe and S. Chiba. *Dynamic Programming Algorithm Optimization for Spoken Word Recognition*, page 159–165. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[34] S. Seabold and J. Perktold. Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference*, volume 57, page 61. Austin, TX, 2010.

[35] K. Shockley, D. C. Richardson, and R. Dale. Conversation and coordinative structures. *Topics in Cognitive Science*, 1(2):305–319, 2009.

[36] A. Sinclair, A. Lopez, C. Lucas, and D. Gasevic. Does ability affect alignment in second language tutorial dialogue? In *19th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2018)*, 4 2018.

[37] A. Sinclair, K. McCurdy, C. G. Lucas, A. Lopez, and D. Gaševic. Tutorbot corpus: Evidence of human-agent verbal alignment in second language learner dialogues. *International Educational Data Mining Society*, 2019.

[38] T. Sinha and J. Cassell. We click, we align, we learn: Impact of influence and convergence processes on student learning and rapport building. In *Proceedings of the 1st Workshop on Modeling INTERPERsonal SynchrONy And infLuence*, pages 13–20, 2015.

[39] D. Tannen. New york jewish conversational style. *International Journal of the sociology of language*, 1981(30):133–150, 1981.

[40] S. Teasley, F. Fischer, P. Dillenbourg, M. Kapur, M. Chi, A. Weinberger, and K. Stegmann. Cognitive convergence in collaborative learning. 2008.

[41] M. Walker and S. Whittaker. Mixed initiative in dialogue: An investigation into discourse segmentation. *arXiv preprint cmp-lg/9504007*, 1995.

[42] A. Ward and D. Litman. Dialog convergence and learning. *Frontiers in Artificial Intelligence and Applications*, 158:262, 2007.

[43] J. T. Webb. Subject speech rates as a function of interviewer behaviour. *Language and speech*, 12(1):54–67, 1969.

[44] D. Weintrop and U. Wilensky. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children*, pages 199–208, 2015.

Using Student Trace Logs To Determine Meaningful Progress and Struggle During Programming Problem Solving

Yihuan Dong
North Carolina State University
ydong2@ncsu.edu

Samiha Marwan
North Carolina State Universtiy
samarwan@ncsu.edu

Preya Shabrina
North Carolina State Universtiy
pshabri@ncsu.edu

Thomas Price
North Carolina State Universtiy
twprice@ncsu.edu

Tiffany Barnes
North Carolina State Universtiy
twprice@ncsu.edu

ABSTRACT

Over the years, researchers have studied novice programming behaviors when doing assignments and projects to identify struggling students. Much of these efforts focused on using student programming and interaction features to predict student success at a course level. While these methods are effective at early detection of struggling students in the long run, there is also a need to identify struggling students during an assignment so that we can provide proactive intervention to prevent unproductive struggle and frustration. This work proposes a data-driven method that uses student trace logs to identify struggling moments during a programming assignment and determine the appropriate time for an intervention. We define a struggling moment as not achieving significant progress within a certain amount of time, relative to the amount of progress made and time taken in a sample student dataset. The paper describes how we determine significant progress and a time threshold for struggling students. We validated our algorithm's classification of struggling and progressing moments with experts rating whether they believe an intervention is needed for a sample of 20% of the dataset. The result shows that our automatic struggle detection method can accurately detect struggling students with less than 2 minutes of work with over 77% estimated accuracy. Our work contributes significantly to building proactive immediate support features for intelligent programming environments.

Keywords

block-based programming, open-ended assignment, struggling, progressing, data-driven method, trace log

Yihuan Dong, Samiha Marwan, Preya Shabrina, Tiffany Barnes and Thomas Price "Using Student Trace Logs To Determine Meaningful Progress and Struggle During Programming Problem Solving". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 439-445. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

1. INTRODUCTION AND BACKGROUND

Computer programming is a challenging topic for novices. As a result, there has been an increasing interest in the early detection of struggling students for proactive intervention to reduce dropout rates and improve student learning in programming courses.

Researchers have collected and studied student problem-solving trace data during programming assignments from various perspectives. Most of this effort has focused on analyzing student programming actions to reveal their behavioral traits and programming patterns. This research typically uses manual inspection of the trace logs [4, 16, 9] or applies machine learning models [7, 3, 1] to categorize student behaviors and discuss the characteristics of each category and their potential impact on or relationship with student performance. The contribution of these studies usually lies in helping educators understand novice learning processes and promote positive behaviors. Another strand of research that analyzed student trace data focuses on student compilation behaviors [11, 19, 2] and syntax errors and bugs [21, 10] in student traces. This research used statistical inferences, machine learning methods, and visualization techniques to explore the relationship between specific patterns and student success and identify novice students' common mistakes. Some of these patterns are helpful for identifying students struggling with certain concepts.

However, we found that there has not been enough research that uses student trace log to model their progress and identify struggling moments during programming assignments. One major application of struggling detection is providing proactive hints in intelligent tutoring systems (ITS), as previous research has shown that novices, especially those with low prior knowledge or experience, may not request on-demand hints even when they need them [18]. Prior work identifying struggling students in traces generally focused on early detection of struggling students determined by the assignment outcome [12, 5, 6] and are not suitable for identifying struggle during programming assignments.

In this work, we propose a novel data-driven approach to

identify struggling moments from student trace data. We describe how we adopted the SourceCheck algorithm to model *student progress* during open-ended programming assignments and how to identify *progressing moments* and *struggling moments* from student progress. We used human experts to evaluate the progressing moments and struggling moments classified by our method. Our initial result shows that human experts had a decent agreement with our algorithm and that it is possible to determine if a student is struggling during programming assignments within two minutes. To the best of our knowledge, this is the first attempt to use a data-driven progress measurement to identify struggling students. We also discuss how our method may generalize to other domains that meet certain requirements.

2. DETERMINE INTERVENTION TIMES

In this work, we consider a student to be *struggling* during problem-solving when they could not make *enough progress* within a *typical amount of time*. As such, to determine the proper time to provide proactive intervention, we need to investigate: 1) how to measure student progress *during* solving a programming assignment, 2) what constitutes significant progress, and 3) how much time it typically takes a student to make significant progress, and finally, 4) what is the appropriate *time threshold* beyond which we consider the student is struggling and need help. This section describes how our algorithm models student progress and identifies potential progressing and struggling moments through these four steps.

2.1 Dataset

We analyzed a dataset collected from an introductory programming course for non-computer science major students in Fall 2017. Students learned to program by doing a series of open-ended programming assignments adapted from the BJC curriculum [8] in a block-based programming environment called Snap*!*.

We extended the Snap*!* environment to record all students' programming actions into *traces*. Each trace, identified by a project id, contains all actions a student performed during an assignment (e.g., creating or deleting a block) with timestamps. There are two types of actions in the traces, *non-coding* actions, and *coding* actions. Non-coding actions do not change the program scripts, for example, searching for blocks and running the program. Coding actions change the abstract syntax tree of program scripts, such as creating variables, creating custom blocks, and reordering blocks. Alongside every coding action, our Snap*!* environment also saves a code *snapshot* after the action, allowing us to reconstruct the steps a student took to build their final code and analyze their coding progress.

In this work, we focused on analyzing two assignments, Squiral and Guessing Game. Squiral is a homework assignment that asks students to create a procedure that draws a square-like spiral with a certain number of rotations specified by an input parameter. A correct Squiral solution typically contains 7 to 14 lines of code. Guessing Game (GG) is an in-class assignment that requires students to create a game that greets the players by their names, asks the player to guess the secret number, and tells the player if their guesses were too high too low until they guessed correctly. A typical Guessing Game solution contains 14 to 18 lines of code. These two assignments allow us to explore how well our method identifies students' struggling moments in assignments with different time constraints. Table 1 shows the descriptive statistics of the traces analyzed in the two assignments. We preprocessed the traces to remove any idle time of more than five minutes, during which the student did not perform any action.

Table 1: Descriptive statistics of the trace logs and grades of the two assignments.

	Traces	Rows	Time on Task	Avg Grade
Squiral	45	25160	29.6m	9.8/12
GG	59	22744	30.5m	11.7/12

2.2 Define Progress

The first step to identify struggling is to measure student progress in the assignment. Previous work used code compilation results [11, 19, 2], students' programming behavior patterns [7], and features completion [15, 13] to monitor student progress in an assignment. While these criteria are reliable indicators of how many assignment requirements the students have met, they did not use the student traces' full potential to identify struggling students at an action-level granularity during the assignment.

We adopted the *SourceCheck* algorithm [17] to measure student progress during an assignment. *SourceCheck* was initially designed as a hint generation algorithm to provide on-demand, next-step hints to help students move towards the closest correct solution to the student's current code. When generating hints, emphSourceCheck first compares the student's current code snapshot with a list of correct solutions (usually collected from past student data) and generates *mapping costs* from the student's code snapshot to each of the correct solutions. These mapping cost values represent how similar the correct solutions are to the student code – the more similar a correct solution is to the student's code, the lower the mapping cost is. *SourceCheck* picks the correct solution with the lowest mapping cost as the *closest correct solution* and generates next-step hints to move the student to that solution.

We adapted the mapping cost into a **similarity score** by reversing the mapping cost such that when a student moves closer to the closest correct solution, the mapping cost decreases, and the similarity score increases. One novel aspect of this paper is how we use the similarity score to measure student progress in the two assignments [1] We calculate a similarity score for every snapshot in a student trace using the *SourceCheck* algorithm. We define a snapshot's *progress* in an assignment as the similarity score *difference* between the *current* snapshot and its *previous* snapshot. As such, at a particular snapshot, we say a student is making a **positive progress** if the similarity score difference is positive and a **negative progress** if the similarity score difference is negative.

[1]Note that we assume a student is moving towards the closest correct solution at any given snapshot. Students do not know the prior student solutions used by *SourceCheck*, and we have no ground truth to identify what strategy a student may be using for their assignments.

We can visualize a student's progress in an assignment by plotting the similarity scores for each snapshot against the cumulative active time, shown in blue in Figure 1. The red line in Figure 1 represents what we call "absolute progress", which we define in the next section. The blue dots in Figure 1 represent the similarity score of every snapshot in a *Squiral* trace, and the blue line represents the progress (similarity change between consecutive snapshots) over time. Figure 1 demonstrates that the student made steady *positive progress* in the first eight minutes. Then, the student had a reduced similarity score for about four minutes, tearing the code apart trying to complete an objective of the assignment. Afterward, the student continued to make rapid positive progress until 14 minutes, stopped progressing for around three minutes, and finally reached their final submission at around 17 minutes.

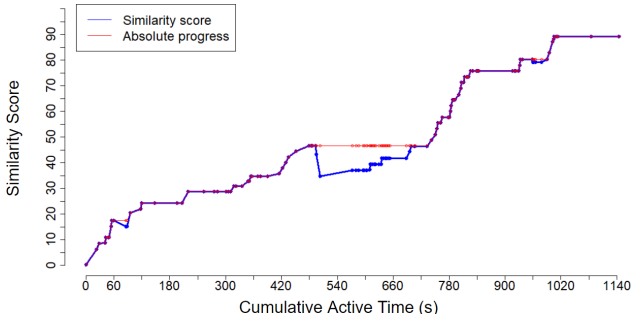


Figure 1: Similarity score (blue) and absolute progress (red) change over time in one student trace of Squiral.

2.3 Determine Significant Progress

Through inspecting multiple students' traces and comparing them with their corresponding progress plot (e.g., Figure 1), we found that not all positive progress represents a significant change to the program. Some minor similarity score increases were due to reordering code that does not change the code semantics but slightly reduces the mapping cost. This observation means that using *any* amount of similarity score increase for making progress *may not* be sufficient. Thus, it is important to determine how much similarity score increase can be considered *significant progress*.

To determine significant progress, we first define *absolute progress* for a student s_j at time t_i. We define $S_{max}(s_j, t_i)$ to be the maximum similarity score achieved by student s_j in an assignment between time t_0 and t_i, $S_{max}(s_j, t_i) = max_{k=0}^{i} S(s_j, t_k)$. We then define **absolute progress** as the difference in the maximum similarity scores between t_i and the previous snapshot time t_{i-1}, $P_{absolute}(s_j, t_i) = max(S_{max}(s_j, t_i) - S_{max}(s_j, t_{i-1}), 0)$. To visualize absolute progress, we plot the highest similarity scores achieved since the beginning of the trace (S_{max}), as shown in red in Figure 1. The absolute progress is *positive* whenever S_{max} increases between two consecutive snapshots.

We then calculated and sorted the absolute progress values from all the student traces for each assignment in increasing order and plotted all *positive absolute progress values* by percentile (using the quantile function in R), as shown in Figure 2. We used the 25th percentile of absolute progress values as the threshold for making **significant progress**. This choice was also used in another work identifying struggling students in a MOOC programming assignments [20]. The intuition is that if a student's absolute progress is no more than three-quarters of all the absolute progress, we consider the student is not making enough progress. Figure 2 shows that the significant progress threshold is 1.25 for Squiral and 1.5 for Guessing Game.

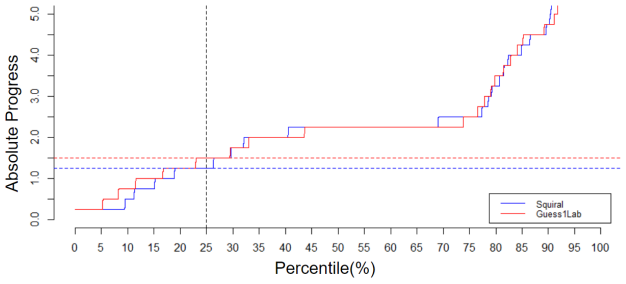


Figure 2: Positive absolute progress in all traces by percentile

2.4 Determine Typical Time

Now that we have determined the significant progress for each assignment, the next step to identify struggling moments is to extract the *typical time* for students to make significant progress. To do this, we first split all the traces into *code chunks* whenever the student makes significant progress–each code chunk contains multiple snapshots. This gave us 648 code chunks from Squiral and 1207 code chunks from Guessing Game. Then, we calculate the elapsed time between the first and the last snapshot in each code chunk and organize the elapsed times in ascending order. We plot the ordered elapsed times in Figure 3 where the y-axis is the elapsed time, and the x-axis is the percentile of the elapsed time distribution. The green line in Figure 3 marks the third quartile of time used to make significant progress. Note that the elapsed time to make significant progress grows almost exponentially after the third quartile. Therefore, we chose to use the third quartile as the cutoff for the *typical time* to make significant progress. The third-quartile time (dashed green line) in Figure 3 intersects with the Squiral progress (solid blue line) at 105 seconds and intersects with the Guessing Game significant progress (solid red line) at 85 seconds. We use these times as the typical time for the students to make significant progress in Squiral and Guessing Game. There are several dashed lines on Figure 3, which we explain in section 3.

2.5 Determine Progressing and Struggling Moments

The last step of this process is to use the typical time to make significant progress in identifying progressing and struggling moments for each assignment. To do this, we took all the code chunks generated in the third step and divided them into two groups, *struggling moments* and *progressing moments*. Recall that we define a student as struggling if the student does not make enough progress within a typical amount of time. Therefore, struggling moments are defined to be code chunks that have elapsed time greater than our struggling time threshold (75th percentile of time for significant progress), meaning that in this code chunk, even though students spent a long time, they did not make

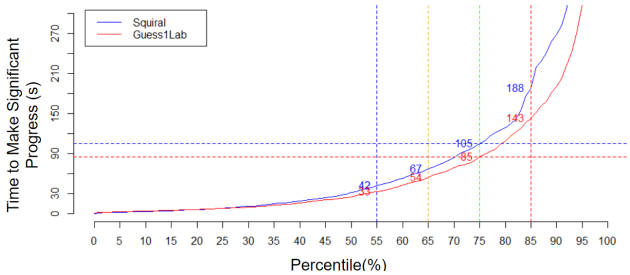


Figure 3: The elapsed time for all code chunks to make significant progress by percentile. The green, yellow, blue and red dash lines mark the proposed, earlier, even earlier, and later intervention times, respectively.

significant progress. Conversely, progressing moments are defined to be code chunks that took equal or less time than the 75th percentile of the typical time to make significant progress. Table 2 summarizes the significant progress, typical time for significant progress, and the number of code chunks generated for each assignment.

Table 2: A summary of products generated by our algorithm for the two assignments.

	Squiral	GG
Sig. Progress	1.25	1.5
Typical Time for Sig. Prog.	105.3s	84.5s
# Code Chunks	648	1207
# (%) Struggling Moments	131(20.2%)	269(22.3%)
# (%) Progressing Moments	517 (79.8%)	938(77.7%)

3. EVALUATION

Our evaluation is driven by the following research questions:

RQ1: To what extent the human experts agree with the progressing moments and struggling moments identified by our method?
RQ2: What are the common causes that the human experts do not agree with the progressing and struggling moments identified by our method?

We invited three expert raters, all included as authors, to participate in the rating of whether an intervention is needed for given code chunk samples. All three experts are computer science graduate students, two with extensive experience analyzing Guessing Game traces, and all three with extensive experience analyzing Squiral traces for research.

We first created the *struggling rating sample dataset* by picking a random struggling moment from each trace until the rating dataset contains 20% of all struggling moments for each assignment. Then, we created the *progressing rating sample dataset* by selecting the progressing moments immediately before each struggling moment in the struggling rating sample dataset. Finally, three redundant progressing moments were excluded from the progressing rating dataset because they were shared by consecutive struggling moments in the same trace. As a result, our rating dataset ended up with 29 struggling moments and 29 progressing moments for Squiral (out of 131), and 57 struggling moments and 54 progressing moments for Guessing Game (out of 269).

The experts were told to imagine themselves as TAs when rating. They used a customized interface that allowed them to visually step through the students' code changes in the rating moments to decide whether an intervention is needed. The experts used the time elapsed between actions and the type of the actions to inform their rating decision. They were asked to avoid using *hindsight*, which means that they should not justify their decision for intervening at an earlier time using student's later actions.

When rating the struggling moments, to make it easy to compare expert ratings, the experts were given five intervention timing options. The five options corresponds to potential intervention times marked by the colored vertical lines shown in Figure 3, which correspond to suggesting an intervention at time: blue or before (quantile(0.55)), yellow (quantile(0.65)), green (quantile(0.75), the typical time to make significant progress), red (quantile(0.85)), or "not now" (after red, or never). We chose these candidate percentiles to gain insights into expert preferences of intervention times for future analysis. For struggling moments, experts were shown the code changes from the start until the last action before the 85th percentile time for significant progress (red) to decide when an intervention would be most appropriate, or "not now." For progressing moments, experts were shown the entire progressing moment (all code changes within the time period where significant progress was achieved) and asked the expert to rate whether the moment "needs intervention" or "not now." Aside from rating the struggling and progressing moments, experts were also encouraged to take notes on why they believed an intervention is needed whenever they rate a sample as "needs intervention" for both the struggling and the progressing rating sample datasets. These notes will help us understand the experts' point of view when inspecting the disagreements between the experts and our algorithm.

Before formal rating, the experts practiced rating on a training dataset by immediately discuss their ratings after rating each sample until they were comfortable with the rating process. Then, the experts rated the struggling moments independently in three rounds, each round rating a third of the samples in the dataset. After each round, the experts gathered and discussed the differences in their ratings to share perspectives and resolve disagreements caused by oversight. We did not require the experts to reach a complete consensus on the rating because experts sometimes have different opinions on handling specific situations. Finally, after rating the struggling sample dataset, the experts independently rated the progressing dataset and were asked to check disagreements to correct rating errors caused by oversight.

4. ANALYSIS AND RESULT

To evaluate our RQ1 considering to what extent the human experts agree with the struggling moments and progressing moments identified by our algorithm, in this analysis, we merged the five rating options the experts used in the rating of the struggling chunks dataset into two options, "need intervention" and "not now." Specifically, we merged "at blue or before," "at yellow," and "at green" options into "need intervention" and merged the "at red" and "not now" options

into "not now." These two option labels are identical to those used when rating the progressing moments to directly compare how much the experts agreed with the generated struggling and progressing moments. Splitting and merging the options at green where the proposed time is at allows us to determine if our proposed typical time to make significant progress is appropriate and enough for the experts to decide whether they believe an intervention is needed.

Our analysis of expert ratings focuses on their ratings after discussion because those ratings are after the effort to remove personal error or bias. However, we report the inter-rater reliability (IRR), calculated with Fleiss Kappa, both before and after discussions to demonstrate the impact of the discussions. To determine how well the experts agree with the algorithm, we turned each expert's ratings into binary scores of 0s and 1s depending on whether the expert rating agrees with the rating dataset. Specifically, for struggling moments, the score is 1 if the expert rated "need intervention" and the score is 0 otherwise. For progressing moments, on the other hand, the score is 0 if the expert rated "need intervention" and the score is 1 otherwise. We then take the average of all expert ratings to calculate a *combined rating score* for each sample such that a combined rating of 0 or 1 represents that the experts reached an agreement and anything in between 0 and 1 means the two or three experts had different opinions, even after discussion, on whether the student was making progress or struggling. Finally, we calculate the *agreement rate* for each rating dataset by summing up all the combined rating scores in that rating dataset and divide the sum by the total number of samples in the rating dataset for that assignment.

4.1 RQ1: Expert Agreement

Table 3 shows the percentage of progressing moments and struggling moments that the expert ratings agreed with the algorithm after discussion, as well as the corresponding inter-rater reliability before and after discussions. Looking at the agreement rate between the expert and our algorithm, we see that for both assignments, over 77% of the time, the human experts agreed that an intervention was needed when the algorithm determined the student was struggling. Over 85% of the time, the human experts agreed that an intervention was not needed when the algorithm determined the student was making progress. This suggests that our method was able to identify struggling moments and progressing moments from the trace data with decent accuracy.

Table 3: Human expert agreement with the algorithm identified struggling moments and progressing moments

	Struggling Rating			Progressing Rating		
	N	Need Interv.	IRR (B/A)	N	Not Now	IRR
Squiral (3 raters)	29	77.0%	0.805** /0.847**	29	85.2%	0.853**
GG (2 raters)	57	83.3%	0.539** /0.646**	54	85.1%	0.819**

Looking at the expert agreement with each other, we found that the experts had excellent inter-rater reliability for progressing moments ratings on both assignments and for struggling moments ratings on Squiral. However, the experts were only able to reach a moderate agreement, even after discussion, for struggling moments on Guessing Game. We explored reasons that might cause experts to disagree with each other by manually inspecting the traces and their notes. We found that in four out of five struggling moments that the experts disagreed, students did not have errors in their snapshots but only performed less than six actions, which is way below the average number of 12 actions of all rated struggling moment samples. In such cases, one expert prefers to hold off on any intervention until seeing more student actions, whereas the other expert believes that the student needed a nudge telling them what they should do next. We did not see such a case in Squiral because all the rated struggling moments with few student actions had relatively apparent flaws in the student codes that warrant intervention. We will talk more about this in the discussion section.

4.2 RQ2: Common Causes of Disagreement

We manually investigated the ratings on which the human tutors and our algorithm disagreed. We present some common causes of disagreement for struggling moments and progressing moments, respectively.

Disagreement in Struggling Moments

Solution Matching: A decreased similarity score does not always mean the student is making negative progress. Due to characteristics of the SourceCheck algorithm, in some cases, a reduced similarity score can also be caused by the SourceCheck algorithm mapping the student's previous snapshot and the current snapshot to different correct solutions because the student added a particular code block. In such cases, if the student similarity score does not surpass the maximum similarity score since the beginning within an expected amount of time, our algorithm will determine the student is struggling, even though the student is making progress (having an increasing similarity score). However, the student may just be using a different approach to solve the problem from the expert's perspective.

Few Coding Actions: Since our algorithm focused on students' progress over time, sometimes students might not have taken enough actions for experts to determine if the student is struggling or not. There are several possible reasons why students have few actions, including trying to reason with their code, running their code, evaluating the result, or being off task. Disagreement in this situation did not only occur between experts and our algorithm but also happened between expert raters, causing a relatively lower inter-rater reliability of struggling moments in the Guessing Game assignment.

Disagreement in Progressing Moments

Logic Errors: The experts are good at catching critical logic errors in the student code and tend to intervene if there is a critical logic error in student code that might prevent them from completing the feature they are working on. For example, in Guessing Game, both experts decided to intervene when a student set the secret number to a boolean value and was trying to use the secret number to give player feedback on their guesses because the student would not be able to test the feedback feature properly without correctly setting the secret number first. In contrast, our algorithm considered that the student was making progress because

the students added blocks seen in the correct solutions.

Human Factors: Sometimes, when deciding on intervention, the experts assess the natural language in students' code to infer information about student intention in a way that is not possible for our algorithm. For example, in the Guessing Game assignment, experts pointed out that an intervention is needed when several students used an if/else block instead of combining the say block and the join words block to greet the players. However, since the if/else block was used for giving player feedback in many correct solutions, our algorithm determined the student was making progress, despite the student using the if/else block incorrectly for a different purpose.

5. DISCUSSION

RQ1: *To what degree do the experts agree with the algorithm on struggling and progressing moments?* For Squiral, the experts agreed that an intervention was needed for over 77% of the struggling moment samples and agreed that no intervention was needed for over 83% of the progressing moment samples. However, the experts have a relatively lower disagreement with each other in Guessing Game ratings, caused by different opinions on whether an intervention is needed when a student has few actions within the time frame. Since the expert raters were not pedagogical experts and had experience level on par with experienced teaching assistants (TAs), we do not know how an experienced instructor would react to this or other scenarios. Nevertheless, our results show that our method of identifying struggling moments by measuring whether a student could make significant progress within a typical amount of time aligns well with opinions from our experts who have at least as much experience as highly qualified TAs. We believe this shows great potential to determine when to give proactive interventions to students in intelligent tutoring systems for programming.

RQ2: *What are the common cases that our algorithm does not handle well?* We listed four common causes that led to disagreements between the experts and our algorithm. The first cause, "solution matching", rests in the adoption of SourceCheck as a progress measure. Since the SourceCheck compares student solutions to multiple model solutions, sometimes, adding a code block to a snapshot could cause SourceCheck to pick a different solution as the closest solution with a lower similarity score. This calls for investigation on how we may wish to smooth out the abrupt progress change when mapped to different correct solutions between consecutive snapshots. "Logic errors" and "human factors" are problems that are difficult to solve by using distance-based similarity measures alone, since similarity measures merely compare the mapping cost between two pieces of code and do not assess the semantics in natural language or programming logic. Thus, these two causes may require obtaining knowledge from other types of analysis to identify. Previous research has used compilation error [11, 19, 2], and feature detection [14] to detect if there are specific errors and determine if the student is struggling. Incorporating these methods could provide richer information in the decision-making for struggling moments. Lastly, some expert disagreements with the algorithm in struggling moments were caused by dealing with "few actions" within the time frame of the code chunks. Our inter-rater reliability for Guessing Game shows that even experts had a hard time agreeing on if an intervention is needed in this case. There seem to be multiple factors that may affect the expert decision, including if there are errors in the snapshot, the type of actions performed, and the assignment requirements that are completed and incompleted. Potential solutions to this problem may include consulting experienced teachers and incorporating sequence pattern analysis [7].

It's worth pointing out one important contribution of this work is that our method of using a data-driven approach to identify struggling moments has the potential to be generalized into any other domain that meets the following criteria.

1.Student Performance: Good student performance on the task, meaning that the majority of the students achieve correct or mostly-correct final solutions.
2.Trace log data: having time-stamped trace logs that documents students' snapshots during an assignment.
3.Progress measure: A score, or a combination of correct solutions and a distance metric between snapshots and correct solutions, as we devised from SourceCheck.

There are two major ways our method can benefit future research. First, our method of identifying progressing and struggling moments provides a new way for researchers to study novice problem-solving behaviors and identify common misconceptions. Second, the significant progress value and typical time to make significant progress can be incorporated into intelligent tutoring systems for providing proactive feedback to struggling students.

This work has some clear limitations. First, we only used three expert raters to evaluate the sample result. The expert raters were not pedagogical experts and had experience levels on par with experienced TAs. Hence the evaluation result may be different if rated by experienced instructors. In addition, our work is limited by a small sample size with only two programming assignments. We need further investigation to determine if our result holds for assignments with even more varied complexity or in other problem-solving contexts.

6. CONCLUSION

This work presented a novel, data-driven approach to use a similarity measure to model student progress in programming assignments and identify progressing and struggling moments from trace log data. To evaluate the performance of our algorithm, we asked human experts to evaluate a sample of 20% of the algorithm-identified progressing and struggling moments from trace logs from students solving two programming assignments and rated if the experts agree with the algorithm. Our result shows that the expert agreed with over 77% of the struggling moments and over 83% of the progressing moments, which shows great potential. Our algorithm can be generalized to different domains if they have good student performance, trace log data, and a progress measure for in-progress student solution attempts.

7. REFERENCES

[1] A. Allevato and S. H. Edwards. Discovering patterns in student activity on programming assignments. In *ASEE Southeastern Section Annual Conference and Meeting*, 2010.

[2] A. Altadmri and N. C. Brown. 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 522–527, 2015.

[3] P. Blikstein, M. Worsley, C. Piech, M. Sahami, S. Cooper, and D. Koller. Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences*, 23(4):561–599, 2014.

[4] Y. Dong, S. Marwan, V. Catete, T. Price, and T. Barnes. Defining tinkering behavior in open-ended block-based programming assignments. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 1204–1210, 2019.

[5] G. Dyke. Which aspects of novice programmers' usage of an ide predict learning outcomes. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 505–510, 2011.

[6] A. Estey, H. Keuning, and Y. Coady. Automatically classifying students in need of support by detecting changes in programming behaviour. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 189–194, 2017.

[7] G. Gao, S. Marwan, and T. W. Price. Early performance prediction using interpretable patterns in programming process data. *arXiv preprint arXiv:2102.05765*, 2021.

[8] D. Garcia, B. Harvey, and T. Barnes. The beauty and joy of computing. *ACM Inroads*, 6(4):71–79, 2015.

[9] R. Hosseini, A. Vihavainen, and P. Brusilovsky. Exploring problem solving paths in a java programming course. 2014.

[10] M. C. Jadud. *An exploration of novice compilation behaviour in BlueJ*. PhD thesis, University of Kent, 2006.

[11] M. C. Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the second international workshop on Computing education research*, pages 73–84, 2006.

[12] Y. Mao. One minute is enough: Early prediction of student success and event-level difficulty during novice programming tasks. In *In: Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019)*, 2019.

[13] S. Marwan, G. Gao, S. Fisk, T. W. Price, and T. Barnes. Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, pages 194–203, 2020.

[14] S. Marwan, G. Gao, S. Fisk, T. W. Price, and T. Barnes. Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, pages 194–203, 2020.

[15] S. Marwan, T. W. Price, M. Chi, and T. Barnes. Immediate data-driven positive feedback increases engagement on programming homework for novices. 2020.

[16] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari. Habits of programming in scratch. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pages 168–172, 2011.

[17] T. Price, R. Zhi, and T. Barnes. Evaluation of a data-driven feedback algorithm for open-ended programming. *International Educational Data Mining Society*, 2017.

[18] T. W. Price, R. Zhi, and T. Barnes. Hint generation under uncertainty: The effect of hint quality on help-seeking behavior. In *International conference on artificial intelligence in education*, pages 311–322. Springer, 2017.

[19] E. S. Tabanao, M. M. T. Rodrigo, and M. C. Jadud. Predicting at-risk novice java programmers through the analysis of online protocols. In *Proceedings of the seventh international workshop on Computing education research*, pages 85–92, 2011.

[20] R. Teusner, T. Hille, and T. Staubitz. Effects of automated interventions in programming assignments: evidence from a field experiment. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, pages 1–10, 2018.

[21] M. N. C. Vee, B. Meyer, and K. L. Mannock. Empirical study of novice errors and error paths in objectoriented programming. In *Proceedings of the 7th Annual HEA-ICS Conference*, 2006.

More With Less: Exploring How to Use Deep Learning Effectively through Semi-supervised Learning for Automatic Bug Detection in Student Code

Yang Shi*, Ye Mao*, Tiffany Barnes, Min Chi, Thomas W. Price
North Carolina State University
Raleigh, NC, USA
{yshi26, ymao4, tmbarnes, mchi, twprice}@ncsu.edu

ABSTRACT

Automatically detecting bugs in student program code is critical to enable formative feedback to help students pinpoint errors and resolve them. Deep learning models especially code2vec and ASTNN have shown great success for *large-scale* code classification. It is not clear, however, whether they can be effectively used for bug detection when the amount of labeled data is limited. In this work, we investigated the effectiveness of code2vec and ASTNN against classic machine learning models by varying the amount of labeled data from 1% up to 100%. With a few exceptions, the two deep learning models outperform the classic models. More interestingly, our results showed that when the amount of labeled data is small, code2vec is more effective, while ASTNN is more effective with more training data; for both code2vec and ASTNN, the more labeled data, the better. To further improve their effectiveness, we investigated the potential of semi-supervised learning which can leverage a large amount of unlabeled data to improve their performance. Our results showed that semi-supervised learning is indeed beneficial especially for ASTNN.

Keywords

CS Education, Machine learning, Program analysis, Bug detection, semi-supervised learning

1. INTRODUCTION AND BACKGROUND

When students encounter difficulties during programming, they are often caused by systemic procedural errors, or "bugs" [9], which can occur repeatedly across problems [38, 8]. For example, a student may confuse when it is appropriate to use the `and` and `or` operators, or fail to consider a boundary case in a condition, using `>` instead of `>=` [17]. These bugs are rarely directly addressed by the compiler or test-case feedback employed in most computer science (CS) courses, which are generally limited to suggesting syntax errors, or which correct input-output pairs the program fails to replicate. Historically, tutoring systems in a variety of learning domains have detected these bugs automatically (e.g. through a *bug library* [9, 4]). The detection can be used to offer tailored formative feedback [34] that address bugs directly [22], and can also help instructors to be more informed about student learning process [25]. The detection of bugs often requires experts' manual definitions, with distinct rules for detecting the bug on different problems [4]. This can make it impractical to use bug detection in practice. Most current automatic grading systems for student code are mainly based on test cases, which provide a score and failed test case information to students [15, 16, 37]. Nevertheless, the relationship between code's output and the presence of specific bugs in student code is not clear, since a given erroneous output could be caused by various errors in student code. An automatic bug detection system for student code could be useful to fill in the gaps for students.

Machine learning (ML) algorithms are powerful tools for data analysis, which have been commonly used for automatic programming code analysis [10]. Classical machine learning methods, such as support vector machines [13] and XGBoost [11], are capable of classifying program code [12, 21, 18]. Recent advances in machine learning have leveraged structural information in code to accurately classify and label it [2, 3, 41, 28]. For example, Alon et al. explored path representations on code represented as trees [2], and designed the code2vec model to learn the representations using deep neural networks [3]. Abstract Syntax Tree based Neural Network (ASTNN) by Zhang et al. applied recursive neural networks in the structure, outperforming Tree-based Convolutional Neural Networks [28] and other state-of-the-art models [41].

However, to apply the models to detect student program bugs, two challenges need to be addressed. First, these deep models were originally designed for *professional programs* which are fundamentally different than code written by students [39]. Some recent work has applied these techniques to educational domains [33, 19, 26, 30, 6], but they either used base models years before, [28, 19], or are not specifically used for bug detection [33, 26, 30, 6]. Second, deep learning models are traditionally "data hungry" [1], using large, labeled

*The first two authors contributed to the manuscript equally.

Yang Shi, Ye Mao, Tiffany Barnes, Min Chi and Thomas Price "More With Less: Exploring How to Use Deep Learning Effectively through Semi-supervised Learning for Automatic Bug Detection in Student Code". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 446-453. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

training datasets (e.g. [19] was trained on 270k samples). However, in most educational settings, datasets can be much smaller (e.g. ∼100 students), and labeling (e.g. to identify bugs) can take extensive expert effort [14]. This suggests the potential of leveraging a *semi-supervised* learning strategy, using a mixture of labeled and unlabeled data [42]. Semi-supervised learning, such as the Expectation-Maximization (EM) method, uses unlabeled data for model improvement [42]. However, studies show that the usage of unlabeled data may not always help [35]. Thus, an empirical evaluation, suggested by recent studies [29], to investigate whether semi-supervised learning with unlabeled data actually helps is needed.

To address these challenges, in this paper, we evaluate two state-of-the-art deep learning methods: code2vec [3] and ASTNN [41], on the task of automatically detecting programming bugs in student code. We manually labeled three bugs in ∼1800 code submissions from 410 students in a Java programming course, where each bug occurred in 4-6 distinct problems. Our results show that, when using *all available training data*, the ASTNN model performs best at detecting all three bugs, outperforming code2vec and two classical baseline models (support vector machines and XGBoost).

Furthermore, we investigate whether a semi-supervised learning approach can improve the code2vec and ASTNN performance without requiring additional labeled data. More specifically, we investigated how the deep and baseline models performed with different amounts of labeled training data through a "cold start" analysis [32]. We found that all models benefited from more data. However, despite deep models' reputation as "data hungry," we found the top-performing model was generally a deep model, regardless of training data size. However, *which* model performed best depended on the data size, with code2vec outperforming ASTNN when less labeled data was available. We also found that semi-supervised learning generally improves both code2vec and ASTNN by using unlabeled data. This effect was most consistent for ASTNN, where semi-supervised learning consistently improved the model performance by 5% to 20% on all splits. For code2vec, we also found that it required very little data (5%) to achieve 80% of its peak F1 score.

The major contributions of this paper are addressing three research questions (RQs):

- RQ1: How well do state-of-the-art deep learning models for programming code perform in a student bug detection task?
- RQ2: How are deep learning models' performance impacted by the amount of available training data?
- RQ3: To what extent does semi-supervised learning improve the performance of the deep learning models?

2. APPROACHES

In this section, we introduce how we build code2vec and ASTNN for program classification; and how we applied the semi-supervised learning strategy on them to leverage unlabeled data.

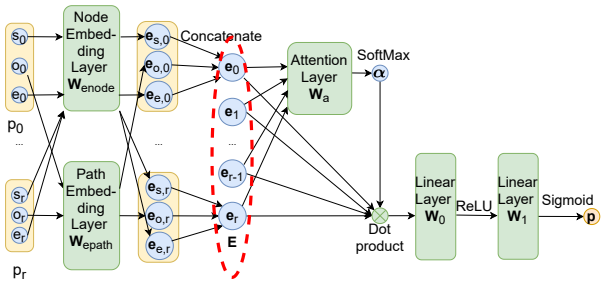


Figure 1: Code2vec model structure: model takes a set of paths as input, and through embedding layers, attention layer, then detect if the input code has bugs (1) or not (0).

2.1 Code2vec

One primary technical challenge in applying machine learning to program code lies in code representation. Code is often represented using an abstract syntax tree (AST) [7], while most learning algorithms expect a fixed-length vector. To solve this issue, sub-components of the ASTs are used as inputs for deep learning models. In the case of the code2vec model, it learns a code embedding through leaf-to-leaf paths, represented as strings. Strings of nodes and paths are mapped into numbers by tokenizers, where different strings are mapped into different numbers. These numbers are used as the input of a code2vec model, shown in Figure 1. Assume we have a code snippet that produces R paths $(p_0, ..., p_r)$ to be fed into the code2vec model. These numbers are embedded into e-dimensional vectors through node and path embedding layers ($\mathbf{W}_{enode}$ and $\mathbf{W}_{epath}$) respectively, and these node and path vectors are concatenated together into one vector for each of the paths $(\mathbf{e}_0, ..., \mathbf{e}_r)$. These vectors form a matrix $\mathbf{E}$, where $\mathbf{E} \in \mathbb{R}^{e \times R}$. Then these path vectors pass through a soft attention layer $\mathbf{W_a}$ [40], where they calculate the soft attention weight α for each of the paths: $\alpha = SoftMax(\mathbf{W_a}^\top \mathbf{E})$, $\mathbf{W_a} \in \mathbb{R}^{e \times 1}$, and thus α has scalar weights α_r for each of the paths, normalized by a SoftMax operator. Then the embedded path vectors $\mathbf{E}$ take the dot product of the calculated attention weights, showing which paths are more important in a code snippet. Then the weighted average vector passes through two fully-connected layers to make the bug classifications. In the training process, all the $\mathbf{W}$ weights are updated using Adam [23] optimization algorithm, while in the evaluation and validation processes, the weights in model are not changed.

2.2 ASTNN

Different from the path-based inputs for code2vec, ASTNN utilize the statement-level ASTs to learn a vector for the code. Specifically, we split the large AST of a code fragment by the granularity of the statement and extract the sequence of statement trees (ST-trees) with a pre-order traversal, and feed them as the raw input of ASTNN. Suppose that we have a set of ST-trees $(s_1, s_2, ..., s_T)$, our goal is to learn a vector representation z for the original code. The detailed architecture of ASTNN is shown in Figure 2.

Statement Encoder: Each ST-tree is composed of a root node and its child indices from a limited vocabulary of up to V symbols. For a ST-tree s_i, we first represent all nodes with

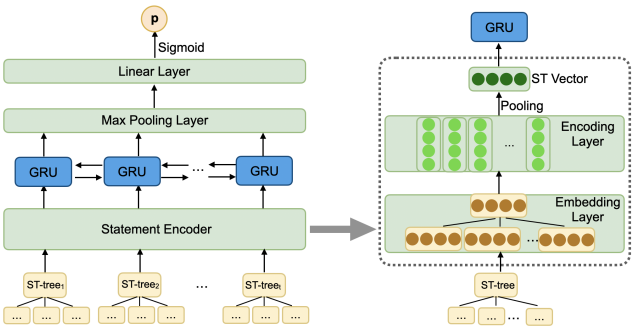


Figure 2: ASTNN model structure: model takes a set of statement trees as input, and through encoder layer, Bi-GRU layer, max-pooling layer, then to detect if the input code has bugs (1) or not (0).

the pre-trained embedding matrix $\mathbf{W}_{embed} \in \mathbb{R}^{V \times d}$ where V is the vocabulary size and d is the embedding dimension. Thus the initial vector of a node n can be obtained by:

$$\mathbf{v}_n = \mathbf{W}_{embed}^{\top} \mathbf{x}_n \tag{1}$$

where x_n is the one-hot encoding of node n. Next the ST-tree will go through a Recursive Neural Network [36] based encoder layer to update the vector for each node:

$$\mathbf{h}_n = \sigma(\mathbf{W}_{encode}^{\top} \mathbf{v}_n + \sum_{i \in child} \mathbf{h}_i + b_n) \tag{2}$$

where $\mathbf{W}_{encode} \in \mathbb{R}^{d \times k}$ is the encoding matrix and k is the encoding dimension. v_n is obtained from Equation 1 and b_n is the bias term. σ is the activation function and in this work we followed the original paper to set σ as identity function. After recursive optimization of the vectors of all nodes in the ST-tree, we sample the final representation e_i for s_i via a max-pooling layer.

Code Representation: Based on the sequences of ST-tree vectors, bidirectional GRU [5] is applied to track the naturalness of statements sequence $(e_1, e_2, ..., e_T)$, where T is the number of ST-trees in the AST:

$$h_i = [\overrightarrow{GRU}(e_i), \overleftarrow{GRU}(e_i)], i \in [1, L] \tag{3}$$

The statement representation $h_i \in \mathbb{R}^{L \times 2m}$, where m is the embedding dimension of GRU. Finally, similar to Statement Encoder, a max-pooling layer is used to sample the most important features on each of the embedding dimensions. Thus we get $z \in \mathbb{R}^{2m}$, which is treated as the final vector representation of the original code fragment. Finally $\mathbf{z}$ vectors pass through a linear layer to make the final classification of the bugs.

2.3 Semi-supervised Learning Strategy

While we explore the potential of machine learning models using insufficient labeled data as training inputs, unlabeled data can also serve as an important resource for the models to learn the structure of code. We applied a semi-supervised learning strategy to utilize these unlabeled data to help the model update. Specifically, in our experiments, we used Expectation-Maximization (EM) method [42] as an exploratory attempt.

EM method is iterative, and it contains two steps for every iteration: 1) In expectation steps, the model infers on the unlabeled dataset, getting a probability score, which will be served as the pseudo-label in the next step, for each of the unlabeled code snippets; 2) In maximization steps, the model is retrained using all the labeled training dataset and the unlabeled set with the pseudo-labels from expectation step. After retraining, the model is used for the next round of expectation step. In our case, deep learning models are designed to output probability scores, but SVM and XGBoost models make classifications without clear scores or probabilities. We implemented the regression versions of the models, assuming they would output a continuous probability as the regression result. We then used 0.5 as the probability threshold to binarize the output, serving classification results. Every model uses a unified 10 iterations of EM steps, assuming the models are able to converge after a certain number of iterations and retraining.

3. EXPERIMENT SETTINGS

3.1 Dataset and Bug Labeling

We performed bug classification on a publicly available dataset, collected from an entry-level Java programming class in Spring 2019[1]. It was collected from the CodeWorkout [15] platform and stored in ProgSnap2 [31] format. Since Java compiler can already detect bugs from code that failed to compile (due to syntax errors), and this code cannot be converted into an AST, we excluded uncompilable code from our analysis. We also did not use code that passed all test cases, as this code is correct and therefore is very unlikely to have bugs. There are 410 students, who attempted in total 50 problems from 5 assignments. Typical solutions for these assignments range from 10 to 20 lines of code. In order to determine the common set of bugs across different problems, two authors examined student code from six distinct programming problems from the first assignment and identified common bugs that arose. They then selected 3 prevalent ones after calculating the coverage of bugs from each problems, and identified in prior CS education literature [17, 20]. This included 2 logical bugs and 1 syntax bug: `comparison-off-by-one` (logical), `assign-in-conditional` (syntax), and `and-vs-or` (logical), defined below:

`comparison-off-by-one`: This bug occurs if, in a conditional expression (e.g. in an `if` or `while`), the student's code uses a greater/less-than comparison operator (`<=`, `>=`, `<`, `>`) incorrectly, and this error can be resolved by adding or removing the '`=`', (e.g. `<` becomes `<=`). The direction of comparison (i.e. `<=` vs `>=`) should already be correct. This often indicates an "off by one" error, and it is contextual, dependent on the number of literals being compared. If there are multiple bugs, including this bug, we still count it.

`assign-in-conditional`: This bug occurs if, in a conditional expression, a student uses the `=` *assignment* operator in their code when trying to compare a variable with another value, rather than the correct `==` *comparison* operator. This is a syntax-based bug, but it is not detected by the compiler, since the assignment is logically a valid operation.

`and-vs-or`: This bug occurs if a student uses the logical

[1]https://pslcdatashop.web.cmu.edu/Project?id=585

Table 1: Detection performance for four classifiers on three bugs.

	Method	Accuracy	AUC	Precision	Recall	F1 Score (Std)
comparison-off-by-one	SVM	0.753	0.658	0.731	0.100	0.173 (0.045)
	XGBoost	0.505	0.541	0.384	0.547	0.334 (0.088)
	Code2Vec	0.736	0.746	0.500	0.556	0.522 (0.058)
	ASTNN	0.785	0.704	0.606	0.533	**0.560 (0.090)**
assign-in-conditional	SVM	0.943	0.959	0.918	0.627	0.733 (0.099)
	XGBoost	0.847	0.877	0.494	0.726	0.563 (0.112)
	Code2Vec	0.917	0.907	0.725	0.688	0.672 (0.119)
	ASTNN	0.970	0.901	0.961	0.807	**0.868 (0.094)**
and-vs-or	SVM	0.722	0.674	0.534	0.173	0.256 (0.078)
	XGBoost	0.503	0.669	0.350	0.784	0.470 (0.045)
	Code2Vec	0.758	0.821	0.570	0.663	0.609 (0.078)
	ASTNN	0.880	0.837	0.820	0.739	**0.773 (0.064)**

operator `and` instead of `or` in their code, or vice-versa, such that the opposite operator would produce correct code. This is also a logical bug that requires contextual information but is easier to detect than `comparison-off-by-one`. It requires the literals, but does not depend much on problem requirements.

Two authors started by labeling 20% of the data, following the same set of initial bug definitions. The labeling process was iterative: the two authors first labeled 20% of the data independently and then calculated Cohen's Kappa scores κ. If on any of the three bugs, the two authors did not achieve a score higher than the 0.8 [24], then the authors discussed and resolved the disagreements, refined the definitions, and continued to another round of independently labeling 10% of the data. This process continued until the authors reached high agreement ($\kappa > 0.8$) on all three categories of bugs, which occurred after labeling 40% of the data. The first two rounds of labeling did not achieve a high κ score, both due to the low scores on the `comparison-off-by-one` bug, suggest that this bug may be more difficult to consistently detect for humans. On the third round of labeling, the two authors achieved 0.81, 0.97 and 0.84 κ scores on the three bugs. Then the authors divided the rest 60% of data by 35% for each person to label, overlapping on 10% of the data for verification. These 10% of data achieved 0.78, 0.98 and 0.95 κ scores, indicating moderate to near-perfect agreement [27]. The finalized labeled dataset has a biased distribution, as only 30% of the submissions have `comparison-off-by-one` bug, 28% of the submissions have `and-vs-or` bug, and 13% have `assign-in-conditional` bug. In total, we spent around 20 hours and labeled 1867 code snippets from 296 students.

3.2 Splits in Experiments

Since our dataset included multiple attempts from a given student, we split our data into training and testing sets by student. This ensured that a given student's code showed up in either the training *or* testing set, but not both. In our experiment, we have 20% of the data as the test set, and the rest 80% are used for model generation. To check the performance of models with limited labeled data, we further split the 80% of data into labeled data and unlabeled data. We use only labeled data for supervised learning, and use both labeled and unlabeled data for semi-supervised setting. All these splits were stratified according to the class label and number of submissions, ensuring that a similar proportion of buggy/non-buggy programs were in each split. This is necessary, since splitting by students can create very biased distributions, especially when we only have small labeled training sets. The stratification uses thresholds for 1) the ratio of bugs and 2) averaged submission numbers for students in respective bug groups. We argue that in practice, we should be able to select a similarly representative sample by manually checking several submissions to see if the distribution is fundamentally different. To ensure we evaluate our model performance with fair comparisons, we created 10 different splits, generated randomly. All models use the same training/testing splits, and average performance metrics are reported as the results. For semi-supervised setting, we varied the size of labeled/unlabeled data to evaluate the performance of models. In order to perform fair comparisons, all semi-supervised models have the same labeled/unlabeled splits. Also, all models are tested on the same test sets, regardless of the model, the amount of training data, or supervised vs. semi-supervised. These settings ensured fair comparisons across different models.

3.3 Model Settings

SVM and XGBoost Parameters: We performed grid search on hyperparameters for SVM and XGBoost models using cross-validation on the training sets. In the SVM setting, we searched linear and Radial Basis Function (RBF) kernels, with C parameters in a range of (0.1 - 1), stepping by 0.1. In the XGBoost setting, we searched through situations that sub-sample portions from 0.1 to 1, stepping by 0.1, using 5 to 100 estimators in the model. To prepare numerical input, we used TF-IDF feature extraction on the code submissions for both models.

Code2vec and ASTNN Parameters: Since deep learning models are more time- and resource-consuming, and our cold start experiments required many repeated runs ($\sim$ 100 runs), we did not perform automatic grid search; rather, we used default settings of the hyper-parameters and did manual changes. In code2vec, after observing the training and validation loss, we set the maximum training epochs as 200, with the *patience* of early stopping set to 100, and set the learning rate to 0.0002. Linear layer and embedding dimensions were kept at the default value of 100. To ensure the highest efficiency of the model, we set the batch size as the full batch. These parameters are tuned with different numbers, but little change in validation accuracy is observed. We also manually padded the number of paths to 100 over all code submissions. In ASTNN, we padded the statement sequences to the maximum length to accommodate the longest sequence before feeding to Bi-GRU. During training, we used 32 as batch size, 0.001 as learning rate, and keep the max training epoch as 50. The encoding dimension for the statement encoder was 128, and hidden neurons for Bi-GRU were 100. The weights were learned during training using the Adam optimizer for code2vec and ASTNN models.

4. RESULTS

4.1 Bug Detection Model Performance

In this subsection we address RQ1: *How well do state-of-the-art deep learning models for programming codes perform in a student bug detection task?* Table 1 shows the results of the classifiers in the task of detecting bugs across problems. We use accuracy, Area-under-curve (AUC), Precision (P), Recall (R), and F1 score as the evaluation metrics for the

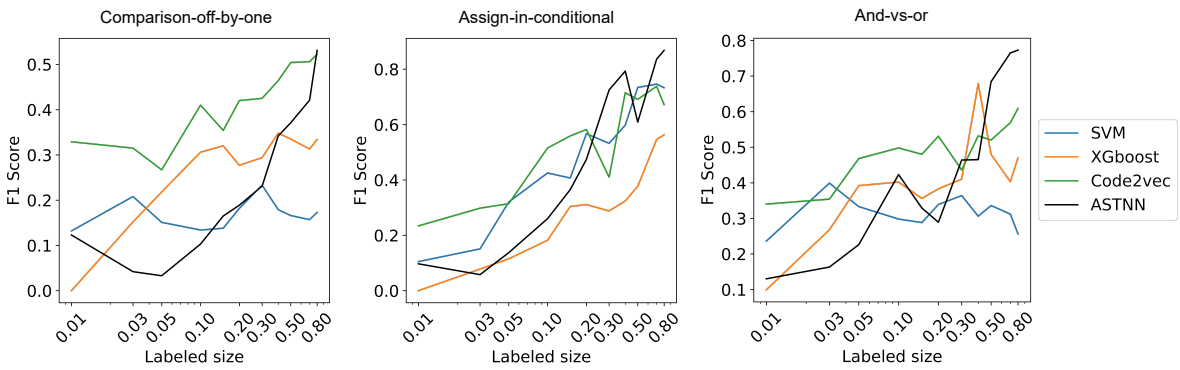


Figure 3: F1 score of models using supervised strategy with different portions of labeled data in detecting bugs.

detection. Across the three bugs for detection, we observe that the top-performing model (ASTNN) achieved 0.560, 0.868, 0.773 F1 scores on detecting `comparison-off-by-one`, `assign-in-conditional`, and `and-vs-or` respectively. The F1 scores indicate that the models achieved a higher performance on detecting `assign-in-conditional` compared to `comparison-off-by-one` and `and-vs-or`. In all bug detection tasks, ASTNN achieved the best F1 score on all bugs. On the two logical bugs, ASTNN achieved at least 0.226 higher F1 score than SVM and XGBoost, while Code2vec also achieved at least 0.139 higher, showing deep learning models are more preferable for detecting the two logic bugs across the six problems. On the detection of the `assign-in-conditional` bug, ASTNN achieved a high F1 score of 0.868, while a simple SVM model is able to achieve a 0.733 F1 score, which is not much lower than the ASTNN model. However, the recall of SVM is low (0.627), which indicates a limited capability of detecting the bugs out of submissions with bugs. Code2vec model did not achieve better F1 or AUC scores than SVM or ASTNN model in this case, showing that in the detection of syntax issues, paths features might be overly complicated. SVM might have a good performance when the real rule to learn is just "If it has `=` instead of `==` in the code, it has the bug," since there is little contextual information to learn. Generally speaking, when using all 80% labeled dataset ($\sim$ 1493 programs on average), deep learning models have a better performance than traditional machine learning models in detecting logical bugs, showing the advantage of leveraging structural information in the feature extraction step.

4.2 Bug Detection with Limited Labels

We address RQ2 in this subsection: *How are deep learning models' performance impacted by the amount of available training data?* From Figure 3, we see the F1 scores of the four models in supervised strategy using a subset of labeled data. The x-axis is the log-scaled labeled data size, and the y-axis is the F1 score that models achieved across the 10 different splits. The lowest portion of labeled data we use is 1%, which contains around 15 students, while the highest portion is 80%. The general trend of the supervised models shows that when more data is used, better F1 scores can be achieved by models, especially ASTNN. We also observe some interruptions in the increment of the performance as more data is available, meaning that it is not guaranteed that more data generate better models. For other baseline models, such a data-performance relationship is weaker, but still more data can generally produce better models.

While the models expect better performance given more data, we would like to note that among all supervised models, code2vec achieved better results than other models using a small subset of labeled data, showing a property of warm starting. With 10 percent of labeled data, code2vec has at least 7.5% higher F1 scores than any other models on all the three detected bugs. When more data is used, ASTNN outperforms other models, showing that there is generally at least one deep learning model more preferable than baseline models. When comparing code2vec with ASTNN, we find that deep learning models are not always "data-hungry": although both models are are sensitive to data size, code2vec starts higher than baseline models in classifying all three bugs. To achieve a good detection result, using 30%-40% (560-747) less labeled data would create models achieving 80% of the F1 score.

With these results we are able to conclude the answer for RQ2: For code2vec and ASTNN, more data would produce models with better performance. However, the relation is not linear: ASTNN is more "data-hungry" than code2vec, but these deep learning models do not require lots of data points to perform better than baselines.

4.3 Application of Semi-supervised learning

This subsection addresses RQ3: *To what extent does semi-supervised learning improve the performance of the deep learning models?* Figure 4 shows the semi-supervised learning results for all four models and the comparisons to supervised ASTNN and code2vec models. The labeled training data for each split is exactly the same as ones used in supervised settings. While the results give a mixed signal about whether semi-supervised learning is beneficial for all models, we have two observations. 1) semi-supervised learning enhanced the learning of *deep models*, especially ASTNN in all three bugs. Comparing the black lines, we found that solid lines are always higher than dashed ones. It suggests ASTNN, as a more "data-hungry model", is favored by the semi-supervised strategy more than in other models. Typically, an ASTNN model trained with a semi-supervised learning strategy achieves 0.05 to 0.2 higher F1 scores than

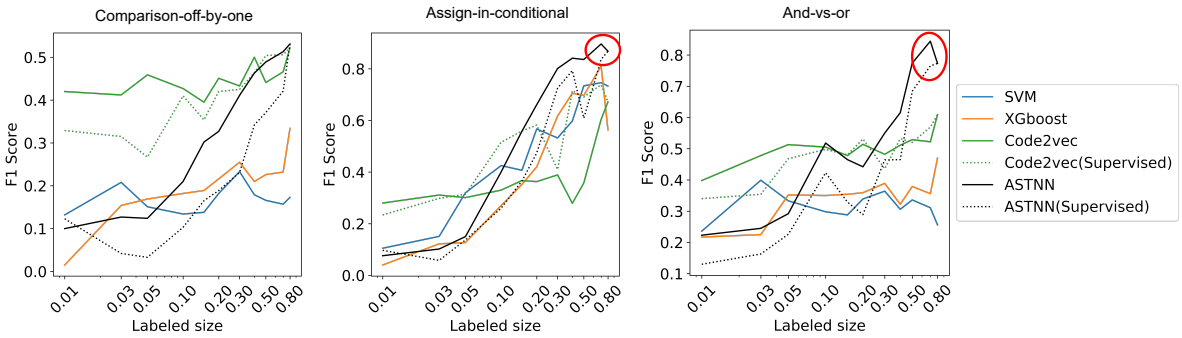


Figure 4: F1 score of models using semi-supervised strategy with different portions of labeled data in detecting bugs. Red circles noted places when semi-supervised strategy outperformed supervised training with full data.

those trained in supervised learning strategy, using the same training dataset. In code2vec, which is also a deep learning model, semi-supervised learning does not always help. It helps code2vec achieve better F1 scores when using a lower portion of data, but when given more data, a supervised learning strategy provides better performance. Semi-supervised learning does not help much for the other two classical models, compared with deep models. 2) In semi-supervised learning scenario, ASTNN achieved a better performance when using 70% labeled data than using all 80% as training in detecting two bugs, `assign-in-conditional` and `and-vs-or`, by 2.8% and 7.1% respectively (red-circled in Figure 4). While this may reflect the fluctuation of data performance, we did run the models 10 times. This suggests that the model may be harnessing the semi-supervised learning strategy to infer labels for unlabeled sets, and achieve more consistent labels than the authors, or some outliers present in the unlabeled set. We assume that the model then learned on these automatically inferred labels and achieved better results than learning from all expert labeled data.

Our conclusion for RQ3 is that semi-supervised learning often improves performance, especially when little training data is available. It enables the models to achieve an expected performance with less labeled data than the supervised scenario. Specifically, semi-supervised learning helped all cases in the learning of ASTNN models, and helped code2vec overall as well, especially when data size is low.

5. DISCUSSION CONCLUSION

Our results suggest three primary conclusions: **1) The two deep learning models generally outperformed baselines, and ASTNN had the best performance.** Our results from Subsection 4.1 show that deep learning models can detect simpler bugs, but still have a limited effectiveness on more complicated bugs (detailed in Subsection 3.1). The complexity of the `comparison-off-by-one` bug may be due to the difficulty of the labeling process, or its dependency on the problem context. **2) Deep learning models may still be successful when labeled data is limited**. From the results in Subsection 4.2, we learn that even if training with small data size such as < 100 data points in complicated programming data, the code2vec model is still able to outperform baseline models. **3) Semi-supervised learning has the potential to help deep learning models perform better.** Semi-supervised learning helped code2vec to achieve a higher performance, but only when a small number of data points are labeled. One may assume the difference between the two deep learning models come from the structures, but it may also come from the feature extraction process. Code2vec uses paths based features but ASTNN uses node based features, and recursively processed by neural networks.

Our results can have other potential applications in educational program analysis tasks as well. For example, as features are automatically extracted from student code during code2vec or ASTNN training, these features can be used to help instructor discover *new bugs*, as suggested by [33], which can help shape instruction. If more features such as problem requirements and test case inputs are available, we can apply these features to the model introduced by [30] to propagate instructor feedback to all students who would benefit form it.

This work also has a couple of caveats or limitations as of the current stage. 1) We only performed extensive experiments on three bugs and used them to generalize to conclusions. This is because the dataset labeling is time consuming, requiring the authors to label $\sim$ 1800 data points. The conclusions here may not generalize to other bugs or code classification tasks. 2) Similarly, these bugs also come from one programming assignment near the beginning of the course, focused on if conditions, and thus may be biased to this specific type of problem. 3) In the splitting process, we performed stratified sampling, requiring that test, labeled, and unlabeled data be a similar distribution of class labels and the number of attempts. 4) Since we only compared our models with two classical model baselines, there may be other better models existing for better performance. We used our best effort to select representative models that achieve state of the art performance, but there might be better models available for the task as well. This work's primary goal is to lay the foundation for using deep models in this task by exploring if the "data-hungry" property also applies here, and potential applications of semi-supervised learning. It serves as a step towards future model designs specific for automatic student bug detection, and provides guideline for situations when labeled data is limited.

Acknowledgements: This research was supported by the NSF Grants: #1623470, #1726550, #1651909 and #2013502.

6. REFERENCES

[1] C. C. Aggarwal et al. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

[2] U. Alon, M. Zilberstein, O. Levy, and E. Yahav. A general path-based representation for predicting program properties. *PLDI'18*, 2018.

[3] U. Alon, M. Zilberstein, O. Levy, and E. Yahav. code2vec: Learning distributed representations of code. *POPL'19*, 2019.

[4] P. Baffes and R. Mooney. Refinement-based student modeling and automated bug library construction. *Journal of Artificial Intelligence in Education*, 7(1):75–116, 1996.

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] A. Bajwa, A. Bell, E. Hemberg, and U.-M. O'Reilly. Analyzing student code trajectories in an introductory programming mooc. In *2019 IEEE Learning With MOOCS (LWMOOCS)*, pages 53–58. IEEE, 2019.

[7] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier. Clone detection using abstract syntax trees. In *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*, pages 368–377. IEEE, 1998.

[8] J. Bonar and E. Soloway. Preprogramming knowledge: A major source of misconceptions in novice programmers. *Human–Computer Interaction*, 1(2):133–161, 1985.

[9] J. S. Brown and K. VanLehn. Repair theory: A generative theory of bugs in procedural skills. *Cognitive science*, 4(4):379–426, 1980.

[10] Y. Brun and M. D. Ernst. Finding latent code errors via machine learning over program executions. In *Proceedings. 26th International Conference on Software Engineering*, pages 480–490. IEEE, 2004.

[11] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4), 2015.

[12] J. Choi, H. Kim, C. Choi, and P. Kim. Efficient malicious code detection using n-gram analysis and svm. In *2011 14th International Conference on Network-Based Information Systems*, pages 618–621. IEEE, 2011.

[13] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[14] A. Dutt, M. A. Ismail, and T. Herawan. A systematic review on educational data mining. *Ieee Access*, 5:15991–16005, 2017.

[15] S. H. Edwards and K. P. Murali. Codeworkout: short programming exercises with built-in data collection. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 188–193, 2017.

[16] S. H. Edwards and M. A. Perez-Quinones. Web-cat: automatically grading programming assignments. In *ITiCSE'08*, pages 328–328, 2008.

[17] A. Ettles, A. Luxton-Reilly, and P. Denny. Common logic errors made by novice programmers. In *Proceedings of the 20th Australasian Computing Education Conference*, pages 83–89, 2018.

[18] A. Gupta, S. Sharma, S. Goyal, and M. Rashid. Novel xgboost tuned machine learning model for software bug prediction. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, pages 376–380. IEEE, 2020.

[19] R. Gupta, A. Kanade, and S. Shevade. Neural attribution for semantic bug-localization in student programs. *Network*, 1(P2):P2, 2019.

[20] M. Hristova, A. Misra, M. Rutter, and R. Mercuri. Identifying and correcting java programming errors for introductory computer science students. *ACM SIGCSE Bulletin*, 35(1):153–156, 2003.

[21] A. Kaur, S. Jain, and S. Goel. A support vector machine based approach for code smell detection. In *2017 International Conference on Machine Learning and Data Science (MLDS)*, pages 9–14. IEEE, 2017.

[22] H. Keuning, J. Jeuring, and B. Heeren. Towards a systematic review of automated feedback generation for programming exercises. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 41–46, 2016.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[25] M. H. Liu. Blending a class video blog to optimize student learning outcomes in higher education. *The Internet and Higher Education*, 30:44 – 53, 2016.

[26] Y. Mao, S. Marwan, T. W. Price, T. Barnes, and M. Chi. What time is it? student modeling needs to know. In *In proceedings of the 13th International Conference on Educational Data Mining*, 2020.

[27] M. L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282, 2012.

[28] L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin. Convolutional neural networks over tree structures for programming language processing. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[29] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *Advances in Neural Information Processing Systems*, 31:3235–3246, 2018.

[30] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas. Learning program embeddings to propagate feedback on student code. In *International Conference on Machine Learning*, pages 1093–1102, 2015.

[31] T. W. Price, D. Hovemeyer, K. Rivers, G. Gao, A. C. Bart, A. M. Kazerouni, B. A. Becker, A. Petersen, L. Gusukuma, S. H. Edwards, et al. Progsnap2: A flexible format for programming process data. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pages 356–362, 2020.

[32] T. W. Price, R. Zhi, Y. Dong, N. Lytle, and T. Barnes. The impact of data quantity and source on the quality of data-driven hints for programming. In *International conference on artificial intelligence in education*, pages 476–490. Springer, 2018.

[33] Y. Shi, K. Shah, W. Wang, S. Marwan, P. Penmetsa, and T. Price. Toward semi-automatic misconception discovery using code embeddings. In *The 11th International Conference on Learning Analytics Knowledge (LAK 21)*, 2021.

[34] V. J. Shute. Focus on Formative Feedback. *Review of Educational Research*, 78(1):153–189, 2008.

[35] A. Singh, R. D. Nowak, and X. Zhu. Unlabeled data: Now it helps, now it doesn't. In *NIPS*, volume 21, pages 1513–1520, 2008.

[36] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, 2011.

[37] J. Spacco, D. Hovemeyer, W. Pugh, F. Emad, J. K. Hollingsworth, and N. Padua-Perez. Experiences with marmoset: designing and using an advanced submission and testing system for programming courses. *ACM Sigcse Bulletin*, 38(3):13–17, 2006.

[38] K. VanLehn. *Mind bugs: The origins of procedural misconceptions*. MIT press, 1990.

[39] S. Wiedenbeck, V. Fix, and J. Scholtz. Characteristics of the mental representations of novice and expert programmers: an empirical study. *International Journal of Man-Machine Studies*, 39(5):793–812, 1993.

[40] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul 2015. PMLR.

[41] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu. A novel neural source code representation based on abstract syntax tree. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 783–794. IEEE, 2019.

[42] X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

Speeding up without Loss of Accuracy: Item Position Effects on Performance in University Exams

Leonardo J. Vida
Utrecht University
Heidelberglaan 8
Utrecht 3584 CS
The Netherlands
l.j.vida@uu.nl

Maria Bolsinova
Tilburg University
Prof. Cobbenhagenlaan 225
Tilburg 5037 DB
The Netherlands
m.a.bolsinova@uvt.nl

Matthieu J.S. Brinkhuis
Utrecht University
Princetonplein 5
Utrecht 3584 CC
The Netherlands
m.j.s.brinkhuis@uu.nl

ABSTRACT

The quality of exams drives test-taking behavior of examinees and is a proxy for the quality of teaching. As most university exams have strict time limits, and speededness is an important measure of the cognitive state of examinees, this might be used to assess the connection between exams' quality and examinees' performance. The practice of randomization within university exams enables the analysis of item position effects within individual exams as a measure of speededness, and as such it enables the creation of a measure of the quality of an exam. In this research, we use generalized linear mixed models to evaluate item position effects on response accuracy and response time in a large dataset of randomized exams from Utrecht University. We find that there is an effect of item position on response time for most exams, but the same is not true for response accuracy, which might be a starting point for identifying factors that influence speededness and can affect the mental state of examinees.

Keywords

exam quality, computerized testing, item response time, item position effect, speededness, speed-accuracy trade-off

1. INTRODUCTION

The quality of standardized high-stakes tests can be seen as a driver of test-taking behaviors and mental states of test takers. The structured format of these tests, with strict time limits and high consequences attached to the test result, lead test takers to a situation in which they feel more or less comfortable [19]. With the introduction and spread of computerized testing in high stakes tests, more data can be collected on high-stakes tests than in the past. These data can be used to monitor the quality of measurement instruments of individual items and exams as a whole. Among the collected data, an important source of information are response times. Response times can be used to gain more information on the test-taking behavior of the examinees [17, 1] and the functioning of the exam and exam questions. As a proxy for exam quality, higher education institutions commonly use reliability measures such as the Cronbach's alpha [5], although literature showed that this indicator, if speededness is present in the exam, might be underestimated leading to reliability concerns [2]. However, reliability is not the only measure of exam quality: test takers behavior, in particular speededness, might be relevant to lecturers and test creators. Thus, investigating the presence of speededness in a test is not only important to know whether the commonly used reliability measure can be trusted, but it can also be used to propose a new indicator of exam quality that takes into consideration the cognitive state of examinees, relating tests' quality and examinees' performance.

Traditional measures of speededness only take into account whether examinees provide responses to all exam questions and are not missing a large proportion of items at the end of the exam [13]. However, fully missing responses at the end of the test is not the only way in which speededness manifests itself [16]. An important way in which time pressure can be observed is the increase of speed and decrease of accuracy close to the end of the test [11]. This behavior can be operationalized as the effect of item position on response time and response accuracy. When exam items are administered to all students in the same order, as often is in the case of traditional high-stakes achievement tests, the effect of item position cannot be separated from the effect of item properties. However, since for test security reasons exams are now more often administered with a randomized item order, it becomes possible to study item position effects separately from item effects.

We have a large data set of computerized exams administered at Utrecht University between January 2015 and June 2020. Using these data, we want to study the overall effect of item position on response time and response accuracy. Furthermore, for each exam we want to quantify the effect of item position on test performance which can be used as an indicator of test quality and of the mental state of test takers. To answer these questions, we focus on three key points. First, we uncover that responses to later items in exams have an increased speed, in conformity with previous studies on anxiety and test strategies within high-stakes

Leonardo Vida, Maria Bolsinova and Matthieu J. S. Brinkhuis "Speeding up without Loss of Accuracy: Item Position Effects on Performance in University Exams". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 454-460. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

tests [19, 8]. Second, we notice the lack of a relationship between item position and accuracy that, if analyzed in parallel with our first finding, might give us indirect evidence that increased response speed does not seem to have the expected negative effect on accuracy. This might show that successful test-taking strategies might include increased response speed towards the end of a test, as previous qualitative analyses already seem to show [18, 10], and that performance-reducing mental states do not appear to influence response speed.

1.1 Problem Statement

Currently, most tests used in European higher education institutions are non-adaptive computerized tests, that often have a large number of multiple-choice items, no penalty for incorrectly responded items and use a test-based time limit instead of a section-based time limit, as is usual in adaptive assessments. High-stakes non-adaptive computerized tests have not been researched and investigated as frequently as their adaptive counterparts and datasets on these tests are not widely available. Thanks to the advance of computerized testing within Dutch higher education institutions, we now have a multitude of data available that were not available before concerning high-stakes tests. Among these data, response times for each examinee on each item and the (randomized) item positions are saved. As test developers, when developing their tests, must find a balance between testing time requirements and difficulty and given that this balance depends on the type of the test and the needs of lecturers and students, we believe that exam-specific effects of item position on response time and response accuracy can provide them with useful information. Therefore, using the dataset at hand we set out to answer the following questions:

1. What is the effect of item position on response time and accuracy?
2. What can be inferred concerning test-taking behavior by analyzing the influence of item position on response time and accuracy?

1.2 Contribution

Answering our research questions, we make several contributions to the field: (1) using generalized liner mixed models that make use of item position in a dataset of university exams, we analyze the effect of item position on response time and accuracy; (2) we provide indications concerning the relation between examinees' response time and response accuracy within randomized high-stakes tests.

This paper is structured as follows: in section 2, we provide the background from which this work stems. Section 3, describes the data and the models used in the analysis. Section 4 continues comparing the results of the models fitted. Section 5 discusses the results of the models and provides the ground for the conclusion in section 6.

2. BACKGROUND

This research revolves around data collected at a higher educational institution concerning the results of tests administered using computers. The computerized collection of students' answers enables the creation of a dataset containing, among other data, the response time data of each student on each test item. We want to make use of this information to help us better understand response processes and, as a consequence, improve measurement instruments.

Interest in response time as a method of revealing information about mental activity has a long origin [15] and other research aims to be relatively comprehensive on the domain [6]. Here, we focus on the specific features on the approach relevant to our data and findings. Recently the role of response time modeling rose to a central position with novels works on the interplay between accuracy and response time [21, 9] and on item position [7]. Traditionally, two main effects of item position have been distinguished: a learning effect when items in later position become easier and a fatigue effect when items become instead more difficult [7]. In both cases, item position effects refer to the impact of the position of an item within an exam on the response time and on the response accuracy. Research commonly assumes that an increase in the speed of response will result in a decrease in accuracy [9]. This relationship, called speed-accuracy trade-off (SAT), is understood as a within-person phenomenon in which the accuracy of response varies with the time taken to produce it [11]. Our empirical findings provide some evidence that might enhance our understanding of the SAT and specify cases in which this relationship is more unclear than what previously thought.

On the other hand, the psychology and education literature has long been interested in developing test designs that generate fair results and thus studied examinees' test-taking behavior to investigate the effect of test designs. Among many domains, this literature also focuses on the effects of anxiety, motivation and test-taking strategies on performance when taking a test [19, 8, 3], finding that high achievers are more likely to engage in effective test-taking strategies compared to low achievers and identifying differences between genders in risk-taking behaviors and anxiety levels. Studies in this area identify risk-taking as an important strategy when taking a test, in particular in multiple-choice tests under strict limits [3]. These guessing strategies are found to potentially lead to better results regardless of ability level and compared to students at the same ability level not using these types of strategies [8]. Our empirical findings provide some evidence also in this aspect, not finding a negative relationship between speeded behavior and response accuracy.

3. METHODS

3.1 Data

We use data from Utrecht University that comprises all exams carried out using the online platform *Remindo Toets*[1] between 2015/01/01 and 2020/06/01. Given our goal of investigating the effect of item position, we select exams in which *response randomization* was applied (i.e., the position of the questions given to examinees changes from examinee to examinee). Therefore, the starting dataset of exams is filtered on the following conditions: (1) duration of the exam: less than 240 minutes. (2) Number of examinees: at least 100. (3) Number of items per examinee: at least 10. (4)

[1] *Remindo Toets* is a software product developed by Paragin, a Dutch education company, which provides educational institution with a platform to create, administer, review and grade exams.

Types of questions in the exam: "choice", "inline-choice", "order", "match". (5) Maximum response time: less than 600 seconds, to reduce outliers in the dataset. (6) Finally, we only analyze exams in which the item order is fully randomized. After filtering, the dataset contains 599.519 item responses. In the final dataset, tests are composed by an average of 204 students, 34 items and the average duration is of 106 minutes.

For each question, lecturers are provided with the so-called p-value, as a measure of "difficulty" of the item, and the estimates of commonly used metrics the item-test correlation (RIT) and the item-rest correlations (RIR) [20]. These variables are available along with item responses and response times. On the dataset at hand, the mean response time is of 91.18 seconds while the average accuracy rate is of 63.19%.

Due to privacy reasons, this dataset was anonymized. Because of this, we are only able to provide a general overview of the exams used in this analysis and not a complete overview of the underlying students population. The dataset at hand consists of 90 unique exams across 6 faculties within Utrecht University. In order, the largest faculties are Science, Veterinary Science and Social Sciences. Finally, as we selected courses with more than 100 examinees and due to the difference in the average class size between bachelor and master courses at Utrecht University, the wide majority of selected exam were from bachelor programs which are typically taught in Dutch. The predominance of the exam in Dutch (90%) indicates that the majority of students attending these courses courses are Dutch, implying that the results of these analyses might be culture-specific.

3.2 Selected variables

In the context of our analysis, we make use of the following variables:

- *Student*: factor variable identifying each individual student. Total number of factors: 18.476.
- *Test*: factor variable identifying each individual test. Total number of factors: 90.
- *Item*: factor variable identifying each individual item. Total number of factors: 5.089.
- *Response time*: continuous variable referring to the total time, in seconds, spent by an examinee on an item. The response time is the summed response time across all attempts made in answering that item. Clear extreme outliers in item were eliminated by setting a cutoff in the filtering process of the data.
- *Accuracy*: binary variable referring to a right or wrong answer by an examinee on an item.

Additionally, we create the following two variables: *item position* and *available time per item*. The first variable, *item position*, is used to identify the location in which the item appears within a test and it is divided within 10 blocks representing 10% of the exam. For each response of person i to item j in exam k, $z_{ijk} \in [0:9]$ denotes the block in which the item was presented to the person. We also create a set of dummy variables $z_{1ijk}, \ldots, z_{9ijk}$, where z_{sijk} if $z_{ijk} = s$, in order to model nonlinear relationship between item position and exam performance. The second variable, *available time per item*, is created dividing the total allotted time for a specific exam by the number of items in that exam. This variable is created as the exams available in the dataset are heterogeneous and do not have a common time limit. As we cannot compare exams having different time limits, we create a variable that represents the time limit at an item level. Across all exams, the *available time per item* has a mean allotted time of almost 3 minutes (177 seconds).

3.3 Models

Before discussing the results of our models, we make a brief note of the reason underlying their creation. A key necessity in our models is the ability to quantify the effect of item position on response time and on response accuracy. Therefore, to build models we turn to generalized linear mixed models (GLMMs). We choose GLMMs as we aim to develop models that create a reliable and easily repeatable analysis to increase the reach and applicability of the model results to datasets from other educational institutions.

In both models, to study the effects on response accuracy and response time, we consider three predictors. We allow for random variability across students incorporating this effect as random effect (θ_{1ik} for the effect on response accuracy, and θ_{2ik} for the effect on response time). We consider fixed item effects (β_{1jk} and β_{2jk} for the effects of item j from exam k on response accuracy and response time, respectively). Finally, for each of the item position dummy variables we consider fixed effects on response accuracy (γ_{1sk}) and on response time (γ_{2sk}), which are estimated for each exam separately. We include fixed item position effects only in the second variation of both models in order to enable us to evaluate whether their addition is significant.

3.4 Modeling response time

We construct the models concerning response time using the logarithm transformation of response time. For response time, we focus on the following linear mixed effect regression ($LMER$) models:

$$\begin{aligned} y_{ijk} &= \beta_{2jk} + \theta_{2ik} \\ \theta_{2ik} &\sim N(0, \sigma^2_{1k}) \end{aligned} \tag{1}$$

$$\begin{aligned} y_{ijk} &= \beta_{2jk} + \sum_{s=1}^{9} \gamma_{2sk} z_{sijk} + \theta_{2ik} \\ \theta_{2ik} &\sim N(0, \sigma^2_{2k}) \end{aligned} \tag{2}$$

wher y_{ijk} is the log-transformed response time of person i on item j in exam k, and σ_{2k} is the variance of the person random effect on response time in exam k. Model 1 does not contain the fixed effects of item position.

3.5 Modeling response accuracy

For response accuracy, we focus on the following generalized linear mixed effect regression on a binary variable (GLMER):

$$\begin{aligned} logit(x_{ijk}) &= \beta_{1jk} + \theta_{1ik} \\ \theta_{1ik} &\sim N(0, \sigma^2_{1k}) \end{aligned} \tag{3}$$

where x_{ijk} denotes response accuracy of person i on item j in exam k, and σ^2_{1k} is the variance of the random effect.

The model is extended with the effect of the item position dummy variables:

$$logit(x_{ijk}) = \beta_{1jk} + \sum_{s=1}^{9} \gamma_{1sk} z_{ijks} + \theta_{1ik} \qquad (4)$$
$$\theta_{1ik} \sim N(0, \sigma^2_{1k})$$

4. RESULTS

We first analyze the results of response time models from equation 1 and 2, before moving to the results of the response accuracy models from equation 3 and 4. Because of limitations due to the size of data at hand, in particular due to the number of fixed effects, all models are fit on each exam individually and their effects are shown as the gray lines in Figure 1 and Figure 2. After fitting each individual model, the item position effects estimates are pooled together with a random-effect meta-analysis in which we the exam-specific effects are assumed to come from a distribution with a common mean and variance [4]. The means of the effect and the ± 1.96 times the standard deviation of the effect boundary are shown as the blue and light blue lines in Figure 1 and Figure 2. The estimates of each of the item position effects of the pooled model of *LMER 2* and *GLMER 4* are given in Table 1 and Table 2 in the Appendix.

4.1 *LMER* on log response time

Concerning *LMER* models on response time, we see that across exams the ANOVAs between model 2 model 1 are significant in 79 out of the 90 cases. Respectively 72 exams at the .1% significance level, 5 at the 1% significance level and 2 at the 5% significance level. The average additional proportion of explained variance of model 2 on model 1 is 0.0084.

Analyzing response time behavior and the effect of item position, we observe a significant negative effect of item position on response time. In particular, we observe an increase in the effect over the course of the exam, which is typically defined as response acceleration. This can be seen in Figure 1 as each item position relates to the effect size compared to the baseline of item position 0, which represents the first 10% of the exam. The blue line represents the pooled estimates of the item position effects across all exams, while the gray lines represent individual exams.

4.2 *GLMER* on response accuracy

With respect to *GLMER* models on response accuracy, we see that model 4 does not significantly outperform model 4. The ANOVAs between model 4 and model 3 are only significant 5 times at the 1% significance level and 11 at the 5% significance level. This shows that the response acceleration found previously and shown in Figure 1 either is not strong enough to influence response accuracy or does not have any influence on it. Hypothesis on the reason underlying this finding are discussed in section 5.

4.3 Proportion of explained variance and available time per item

After running a likelihood-ratio test on each exam individually, we select model 2 as significantly better than model 1 while we further investigated the additional proportion of explained variance of model 2 regressing it on the *available time per item* for each exam. Figure 3 shows that there is a relationship between the available time per item in an exam and the additional explained variance from the model with item position effects (linear correlation $-.37$). This is an indicator that on exams that have less time available, response acceleration is indeed happening and the inclusion of a variable to take into account the position of the item help us explain better the behavior of students. However, as it is visible in Figure 3, the additional explained variance of model 2 is relatively low. This result is important as it provides us with a tool to support the results of response acceleration.

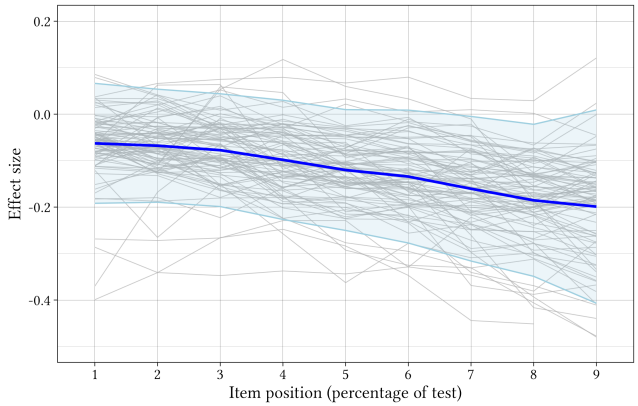


Figure 1: Item position effects on response time

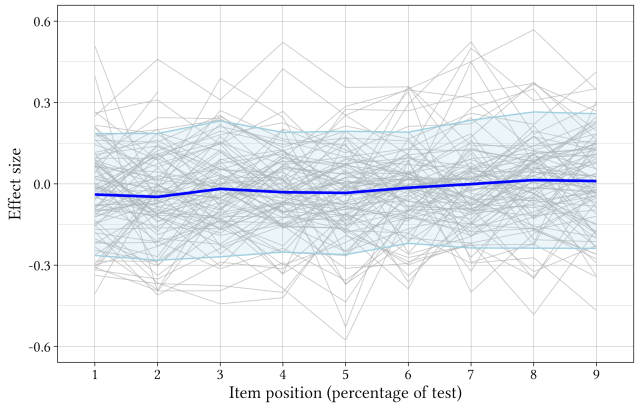


Figure 2: Item position effects on accuracy

5. DISCUSSION

5.1 Discussion of findings

Using a collection of item responses and response times from higher education tests during a five-years time span, we analyze the relationship between item position and response time and between item position and response accuracy. We show that item position is associated with response acceleration while we find that the connection between item position and response accuracy is unclear. Finally, we also find that the available time per item is negatively correlated with the additional explained variance when comparing the model relating item position and response time.

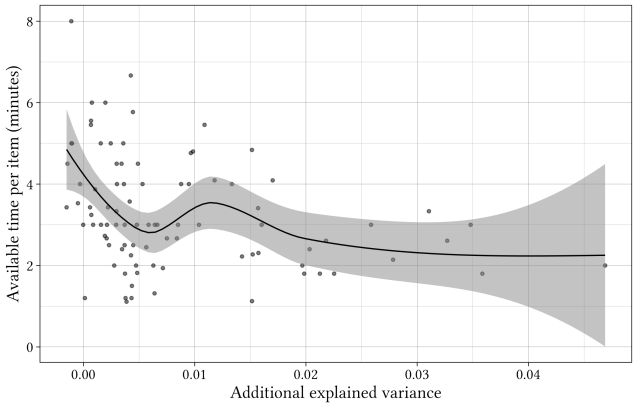


Figure 3: Regression between mean duration of item and additional explained variance for model 2

Concerning the significant effect between item position and response times, we see that the effect is not as strong as we initially expected. The presence of this effect might stem from increased respondents' fatigue or decreased interested in the exam, as highlighted by previous work on this topic [22, 9]. However, previous literature focuses on adaptive tests in which items becomes more or less hard in relation to respondent's performance, while in our dataset this is not the case. Comparing the differences between these two modalities of testing and understanding the difference in the response behavior might be an attractive opportunity for future research.

With regards to the interaction between response accuracy and item position, literature tends to show that later items are more difficult [14, 22] and therefore might decrease the rate of correct responses. We find no significant relationship between response accuracy and item position. The lack of this relationship might be explained by a few hypotheses. First, we might hypothesize that the response acceleration shown is the representation of students reaching their natural speed on the exam. Students might take some time to enter in the mental state that allows them to take an exam and they might be initially slowed down by the need of understanding how the exam is structured. This might explain the increased acceleration of response at the same level of accuracy between earlier and later item in the test. Secondly, we might hypothesize that there is an negative effect of response acceleration on response accuracy, but it is not shown because of the relatively heterogeneous dataset of exams at hand and because of the need of accessing more data. Concerning the second hypothesis, more work on a larger dataset of similar exams is needed before drawing any conclusion. Finally, the lack of this relationship might also be explained by the presence of increased response speed within effective test-taking strategies among high achievers, as found by previous qualitative literature on this topic [18, 12]. This result might be caused by an increased willingness to guess using effective elimination processes, leading to an effective guessing strategy on high-stakes multiple choice tests, when compared to the choice of picking an answer option at random.

Finally, we demonstrate that there is a relationship between the additional explained variance and the available time per item. When the model with item position effects is significantly more informative than the model without these effects, this correlation might also provide backing to a potential quality indicator to be provided to lecturers and test creators to inform them about their tests. In the presence of high additional explained variance, an indicator that would take this relationship into account might be used to provide lecturers with information about the quality of their tests and to identify exam-specific factors that influence speededness and the test-taking strategies of examinees.

5.2 Limitations

When carrying out the modeling part of our research, due to the size of data and factors at hand, we realized that the current statistical methods available to analyze this quantity of data create computational problems. As a matter of fact, we were interested in analyzing the effect of item position on response time and accuracy across the entire dataset but, due to computational limitations, we decided to fit the models on individual exams and later pool the effects estimates using a meta-analysis. To avoid this obstacle, two parallel path might be taken: (1) extending the current statistical libraries to include the possibility of using sparse matrices in computing fixed effects estimates and (2) adding more computational power to the tools used in the analysis.

Further, we also need to take into consideration both the dataset used in this research and the filtering actions taken on it. First, the dataset stems from a higher education institution (*wetenschappelijk onderwijs*) and therefore the results of our analyses might be dependent on the educational level of the students' population. Secondly, because of the filtering actions carried out on the dataset (2), we can assume that Dutch students are more represented in the dataset than international students. This might imply that the results stemming from our analyses are highly dependent on the Dutch test-taking "culture".

An important distinction between our work and previous studies on speededness, such as [9], is that we attempted to remove the effects of very long answers, but not of answers given during *rapid guessing behavior* [15]. As a matter of fact, the validity of test scores of such tests are threatened by what [23] call *noneffort*, which is associated with the guessing behavior of an examinee who does not try to solve items. The effect of such mechanism is an underestimation of his or her actual level of proficiency, threatening the validity of test score by adding a source of construct-irrelevant variance [23]. An analysis about the presence and the methods to identify *noneffort* in this dataset might reveal pathways to either take it into account or eliminate it from the dataset, paving the way for an analysis that compares the estimates of ability accounting only for *truthful* response acceleration.

5.3 Future research

Expanding dataset. As noted in section 5.2, we believe expanding the dataset at hand, including most recent data, and including other universities, would help providing more information on the relationship between response time and accuracy and thus the creation of more accurate indicators for lecturers. Moreover, as there might be differences in

the "culture" of examination between countries and between educational levels, a larger dataset comprising of multiple countries and different educational levels might help clarifying these questions.

Creating a metric to provide recommendations. As we did not see a clear effect of item position on accuracy, a future path for research might assume that, in the presence of similar results, students are given too few items for the time allotted on a specific exam. This would open a path for creating experiments to increase the total number of items on an exam and analyze the cutoff value at which effects on accuracy start appearing. An experiment increasing the number of test items would not only clarify the values at which effects appear, but would also improve domain coverage for that specific exam and, improving its reliability metric, and would allow us to study changes in test-taking strategies within the test.

6. CONCLUSION

This research evaluates methods to investigate the effects of item position on response time and accuracy. We find that, thanks to the advancement in the technologies used in exam settings and the wide application of these technologies in high-stakes tests, these analyses are not only feasible but are also promising if applied on larger datasets. We believe the overall results of the models can be of use within the educational sector, in particular thanks to the creation of an additional and reliable indicator of the quality of an exam. Using the results presented in section 4, we can now answer our research questions:

1. We find a small but significant effect of item position on response time, while we fail to find any effect of item position on accuracy. Section 5 provides a discussion on the findings.

2. We believe our results stemming from modeling item position and response time and the subsequent regression of the additional explained variance of this model and the available time per item, could evolve into a practical indicator providing more information concerning the quality of a test and the test-taking behavior of students.

7. REFERENCES

[1] A. P. Association. Standards for educational and psychological testing, 2014.

[2] Y. Attali. Reliability of speeded number-right multiple-choice tests. *Applied Psychological Measurement*, 29(5):357–368, 2005.

[3] K. Baldiga. Gender differences in willingness to guess. *Management Science*, 60(2):434–448, 2014.

[4] M. Borenstein, L. V. Hedges, J. P. T. Higgins, and H. R. Rothstein. *Introduction to Meta-Analysis*, volume 8. John Wiley & Sons, Bridgewater, NJ, USA, jan 2009.

[5] L. J. Cronbach. Coefficient alpha and the internal structure of tests. *psychometrika*, 16(3):297–334, 1951.

[6] P. De Boeck and M. Jeon. An overview of models for response times and processes in cognitive tests. *Frontiers in psychology*, 10:102, 2019.

[7] D. Debeer and R. Janssen. Modeling item-position effects within an irt framework. *Journal of Educational Measurement*, 50(2):164–185, 2013.

[8] H. Dodeen. Assessing test-taking strategies of university students: developing a scale and estimating its psychometric indices. *Assessment & Evaluation in Higher Education*, 33(4):409–419, 2008.

[9] B. Domingue, K. Kanopka, B. Stenhaug, J. Soland, M. Kuhfeld, S. Wise, and C. Piech. Interplay between speed and accuracy: Novel empirical insights based on 1/4 billion item responses, Mar. 2020.

[10] A. P. Ellis and A. M. Ryan. Race and cognitive-ability test performance: The mediating effects of test preparation, test-taking strategy use and self-eff icacy. *Journal of Applied Social Psychology*, 33(12):2607–2629, 2003.

[11] R. P. Heitz. The speed-accuracy tradeoff: history, physiology, methodology, and behavior. *Frontiers in Neuroscience*, 8:150, 2014.

[12] E. Hong, M. Sas, and J. C. Sas. Test-taking strategies of high and low mathematics achievers. *The Journal of Educational Research*, 99(3):144–155, 2006.

[13] Y. Lu and S. G. Sireci. Validity issues in test speededness. *Educational Measurement: Issues and Practice*, 26(4):29–37, nov 2007.

[14] G. Nagy, B. Nagengast, M. Becker, N. Rose, and A. Frey. Item position effects in a reading comprehension test: an irt study of individual differences and individual correlates. *Psychological Test and Assessment Modeling*, 60(2):165–187, 2018.

[15] T. C. Oshima. The effect of speededness on parameter estimation in item response theory. *Journal of Educational Measurement*, 31(3):200–219, 1994.

[16] D. L. Schnipke and D. J. Scrams. Modeling item response times with a two-state mixture model: A new method of measuring speededness. *Journal of Educational Measurement*, 34(3):213–232, 1997.

[17] D. L. Schnipke and D. J. Scrams. Exploring issues of examinee behavior: Insights gained from response-time analyses. *Computer-based testing: Building the foundation for future assessments*, 34:237–266, 2002.

[18] T. Stenlund, H. Eklöf, and P.-E. Lyrén. Group differences in test-taking behaviour: An example from a high-stakes testing program. *Assessment in Education: Principles, Policy & Practice*, 24(1):4–20, 2017.

[19] T. Stenlund, P.-E. Lyrén, and H. Eklöf. The successful test taker: exploring test-taking behavior profiles through cluster analysis. *European Journal of Psychology of Education*, 33(2):403–417, 2018.

[20] U. University. Toetsanalyse in remindo v2.1. `https://remindo-support.sites.uu.nl/wp-content/uploads/sites/79/2019/11/Toetsanalyse-in-REMINDO-v2.1-003.pdf`, Nov. 2019. Accessed: 2020-10-30.

[21] P. W. van Rijn and U. S. Ali. A generalized speed–accuracy response model for dichotomous items. *psychometrika*, 83(1):109–131, 2018.

[22] S. Weirich, M. Hecht, C. Penk, A. Roppelt, and K. Böhme. Item position effects are moderated by changes in test-taking effort. *Applied psychological measurement*, 41(2):115–129, 2017.

[23] S. Wise and C. DeMars. Examinee noneffort and the validity of program assessment results. *Educational Assessment*, 15(1):27–41, 2010.

APPENDIX
A. ESTIMATES OF POOLED MODELS

Table 1: Coefficients estimates of pooled model *LMER 2*

Item Position	Estimate	Std. Error	Estimate Rand.	Std. Error Rand.	tau
1	-0.0539	0.0036	-0.0628	-0.0628	0.0658
2	-0.0625	0.0035	-0.0679	-0.0679	0.0621
3	-0.0775	0.0036	-0.0775	-0.0775	0.0619
4	-0.0949	0.0035	-0.0983	-0.0983	0.0655
5	-0.1169	0.0035	-0.1203	-0.1203	0.0663
6	-0.1309	0.0035	-0.1343	-0.1343	0.0729
7	-0.1547	0.0036	-0.1605	-0.1605	0.0794
8	-0.1750	0.0035	-0.1854	-0.1854	0.0835
9	-0.1879	0.0034	-0.1989	-0.1989	0.1062

Table 2: Coefficients estimates of pooled model *GLMER 4*

Item Position	Estimate	Std. Error	Estimate Rand.	Std. Error Rand.	tau
1	-0.0412	0.0156	-0.0393	-0.0393	0.1148
2	-0.0412	0.0151	-0.0480	-0.0480	0.1193
3	-0.0163	0.0155	-0.0185	-0.0185	0.1279
4	-0.0283	0.0152	-0.0308	-0.0308	0.1127
5	-0.0264	0.0154	-0.0335	-0.0335	0.1161
6	-0.0106	0.0152	-0.0144	-0.0144	0.1046
7	0.0007	0.0154	-0.0009	-0.0009	0.1203
8	0.0141	0.0152	0.0141	0.0141	0.1282
9	0.0157	0.0147	0.0103	0.0103	0.1268

Grouping Source Code by Solution Approaches — Improving Feedback in Programming Courses

Frank Höppner
Ostfalia University of Applied Sciences
f.hoeppner@ostfalia.de

ABSTRACT

Various similarity measures for source code have been proposed, many rely on edit- or tree-distance. To support a lecturer in quickly assessing live or online exercises with respect to *approaches taken by the students*, we compare source code on a more abstract, semantic level. Even if novice student's solutions follow the same idea, their code length may vary considerably – which greatly misleads edit and tree distance approaches. We propose an alternative similarity measure based on *variable usage paths* (VUP), that is, we use the way how variables are used in the code to elaborate code similarity. The final stage of the measure involves a matching of variables in functions based on how the variable is used by the instructions. A preliminary evaluation on real data is presented.

Keywords

source code distance, semantic analysis, variable usage paths

1. INTRODUCTION

Learning a programming language requires a lot of practice. Students gain knowledge and experience from both, homework and in-classroom exercises. It is, however, very important to give students feedback, e.g. discuss different solution approaches with them. Both, those who have and have not yet succeeded, will benefit from such discussions, either it widens their view (because they may not have thought about alternative approaches yet) or encourages them to try the exercise at a later point in time once more (once they got a glimpse on how to solve it). The sharing of different solution approaches and discussing the pros and cons of different approaches improves their algorithmic thinking skills. However, it is quite common in many programming courses, that the only (automatic) feedback consists of the number of passed or failed unit tests. Achieving a positive feedback then requires an already well developed solution and no credit is given for, say, getting the code structure right – which may frustrate novices noticeably.

Frank Höppner "Grouping Source Code by Solution Approaches — Improving Feedback in Programming Courses". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 461-467. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

We thus address the following research question: Can we support the lecturer in providing meaningful feedback about the solution approaches taken by the students? This requires automation as a lecturer does not have the time to review all solutions manually, especially not in online teaching situation. Providing meaningful feedback requires some kind of insight in the solution approach a particular student was following (semantics), how common the approach is, how many different approaches have been followed in the course, etc. We intend to achieve this by providing a similarity measure for source code that does not focus on results (as unit tests do) but to reach a higher semantic level by assessing the more abstract code structure: different solution approaches manifest themselves in different code structures.

Such a measure would be useful in many ways. For instance, during an in-classroom teaching situation it would enable the lecturer to pick two (or more) submissions following different approaches to start a discussion about their pros and cons. It could also support a lecturer in browsing the spectrum of approaches that were followed by the course members (how many different approaches were chosen how often) without having to check every solution manually. It may enable the lecturer to pick a solution that follows the same approach as a solution that has been discussed already, but did not pass all unit tests, thereby representing a live challenge to the course members ("spot the error"). Note that we are not aiming at grading the source with respect to its correctness, but leave this to the unit tests. As tests do not provide any useful feedback to students that do not yet have an appropriate code structure, the desired measure may close this gap, as it would allow us to differentiate code submissions that are far from working from those that follow a reasonable solution approach and got the code structure right, but only the tiny details prevents them from passing the tests. Such a more detailed inspection would enable a much more sensitive (automatic) feedback.

2. RELATED WORK

Similarity or distance measures for source code have been investigated for a long time. Many approaches have in common that they start from an abstract syntax tree (AST) that represents the code. Code is then compared by calculating some kind of *tree distance* on the corresponding ASTs: the minimal number of steps (node deletion, insertion, and relabelling) to transform one AST to the other. A survey on tree edit distance can be found in [2]. A tree, however, has no conscience about *variable identity*: the very same vari-

able occurs over and over again as a new node in an AST. To reflect which variable change affects other code, program dependency graphs (PDG) may be used. But comparing graphs is even more complicated than comparing trees, a survey on graph edit distance is given in [6]. To simplify the comparison, the AST may also be linearized such that much simpler string comparison measure (edit distance) may be applied (cf. [5, 7, 9]).

An extremly fast approach is to define a hash function on (possibly pre-processed trees) such that a similarity is detected when hash codes are identical [1]. The flexibility of such an approach is, however, very limited, as the similiarity measure is a binary one. It may nevertheless be useful for clone detection (plagiarism) or may be enhanced by calculating multiple hashes [4]. Tree edit distance has been used in, e.g., [10] to detect similar source codes. In [3] a distance measure for source code has been proposed, that transforms the AST into a sequence of tokens on which Levenshtein or edit distance is applied.

In many related papers the goal is to compare (student's) code to a (lecturer's) reference code; the higher the similarity, the closer is the student's submission to the correct solution. In this work, however, we want to identify source codes that are semantically similar, that is, they follow the same solution approach. Especially when novices start to code, their solutions are often more lengthy and redundant than those of an experienced programmer. This affects measures such as edit distance or tree distance dramatically. The approach in [3] did not work out as well as expected, and the authors hypothesized about one of the reasons that the length of the code dramatically influences the edit distance. While longer code may be less readable and more redundant, it is neither more likely wrong nor does it necessarily follow a different solution approach. As an example, consider codes (a) and (b) from Fig. 1. The task was to calculate the mean of all positive array elements. Both codes follow the same approach, but (b) uses a single loop instead of two and is thus more compact. But semantically, we consider both solutions as being identical. We are not aware of any source code similarity measure that tries to reflect that.

It is also common in the literature to replace variable names by a constant string (possibly depending on the variable's type), which eases matching sources that use different variable names. In plagiarism detection (see [5, 9] and references therein) this is considered as a countermeasure against disguising plagiarism by variable renaming. However, the way a single variable is used throughout the code is crucial for a solution approach. Fig. 1(d) utilizes the same statements (e.g., array-access in conditional statement of a for-loop), but does not calculate anything meaningful and thus does not represent the same solution approach as codes (a-c).

3. COMPARING SOURCE CODE BY VARIABLE USAGE PATHS

We argue that two solutions follow the same approach if they use variables in the same way. In the subsequent sections we show how we grasp variable usage in the code and then define similarity measures on this representation.

```
public double avg(double a[]) {
  if (a==null) return NaN;
  int n = 0;
  for (int j=0;j<a.length;++j) {
    if (a[j]>0) ++n;
  }
  double sum = 0;
  for (int j=0;j<a.length;++j) {
    if (a[j]>0) sum+=a[j];
  }
  return sum/n;
} // (a) [TwoLoopQuickExit]
```

```
public double avg(double a[]) {
  if (a==null) return NaN;
  int n = 0;
  double sum = 0;
  for (int i=0;i<a.length;++i) {
    if (a[i]>0) {
      ++n;
      sum+=a[i];
    }
  }
  return sum/n;
} // (b) [OneLoopQuickExit]
```

```
public double avg(double a[]) {
 if (a!=null) {
   int n = 0;
   double sum = 0;
   for (int i=0;i<a.length;++i) {
     if (a[i]>0) {
       ++n;
       sum+=a[i];
   } }
   return sum/n;
 } else {
   return NaN;
} } // (c) [OneLoopGuardedIf]
```

```
public double avg(double a[]) {
  if (a==null)
    return NaN;
  int n = 0;
  double sum = 0;
  for (int i=0;i<a.length;++n) {
    if (a[i]>0) {
      ++i;
      a[i]+=sum;
    }
  }
  return sum;
} // (d) [Disordered]
```

Figure 1: Responses to a simple programming exercise: Write a function avg that yields the mean of all positive values in the array. Left: Code (a) uses two loops, while code (b) only one. While the main code is organized *after* an if-statement in (a) and (b), it is embedded in the conditional statement in (c). Code (d) uses similar instructions, but the variable usage is screwed up and does not solve the problem.

3.1 Variable Usage Paths

Solving a programming exercise requires to combine programming instructions such that a handful of variables jointly build up the final result (and return it to the caller). The key to a solution are thus the variables and how they are embedded in the (possibly nested) instructions. Code analysis often starts with an abstract syntax tree (AST); there, every variable usage is represented by a new node in the tree. This occludes important information: Where in the source code is the same variable used? We therefore rearrange the AST to a graph, where a node is unique for each variable and subsequent usages of the variable link to the same node. Fig. 2 shows an example for the code of Fig. 1(b). All variables are marked in red color. From the paths between the variable `sum` and function `avg`, we can see that the variable `sum` is declared, occurs in a conditional statement that is embedded in a loop, and finally occurs in the return value. We consider these paths (shown in blue in Fig. 2) as a kind of *fingerprint* for the role of this variable: code following the same approach requires variables with the same roles.

We thus do not operate directly on the graph, but paths from variable nodes to the enclosing function node. After applying some transformations to simplify the graph somewhat (e.g. removing *body* or replacing *for*, *while*, etc. by a subsuming label *loop*), we end up with string representations of the three blue paths like *sum/expression/return/avg*, *sum/dec-*

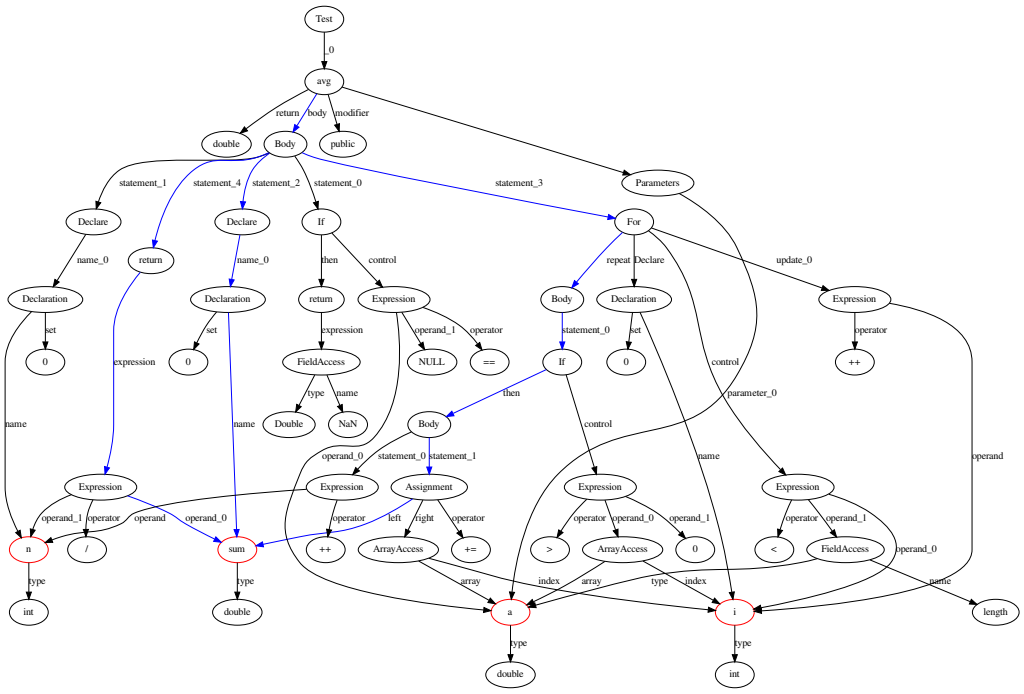


Figure 2: Graph representation of code in Fig. 1(b) (red: variable nodes; blue: VUPs of variable 'sum').

laration/declare/avg and *sum/assignment/if/loop/avg*.

Definition 1. Let I be a set of code instruction labels (such as *if*, *loop*, ...). For a given source code C (single class), let V_C be a set of variable-identifiers[1] and F_C the set of methods declared in C. A **path** $p = (v, i_1, \ldots, i_n, f) \in V_C \times I^* \times F_C := \mathcal{P}$ reflects that a variable v is used by instruction i_1, which is itself used by instruction i_2, etc., in function f. The **VUP-representation** (variable usage path) of code C is a set $P_C \subseteq \mathcal{P}$ of all paths occurring in C.

We thus intentionally drop many details from the original source code, e.g. numerical constants or full expressions. From the fact that two codes that are structurally identical (same VUP representation) we cannot conclude anything about the number of unit tests both codes may pass. But as we have mentioned earlier, we seek for a common code skeleton, which may indicate that the programmers were guided by the same underlying idea. The skeleton includes the information which variables need to be used in which instructions — but there are many different ways of coding expressions equivalently, so we simply stick with unit tests to check their correctness.

3.2 Simplified Set Similarity

Given two source codes (a) and (b) from Fig. 1 and the corresponding VUP-representations P, Q. Code (a) determines the number of elements first before it sums the relevant values, while code (b) does both in a single loop. While the choice of one or two loops may affect the efficiency, they are semantically equivalent and thus *similar*. At first glance the variable usage paths (of, say, the loop counter) in all loops seem identical such that a set comparison (where duplicate elements are not counted) appears just right. However, the loop counter is named `j` in code (a) and `i` in code (b). Furthermore, (a) defines two variables with the same name `j` (one for each loop). Let us ignore these details by introducing a simplification (and revisit the problem later):

Definition 2. From a VUP representation P we obtain a **SVUP representation** (simplified VUP) representation P' by replacing all variable identifiers and all method identifiers by a constant identifier `vn` (generic variable name) and `fn` (generic function name), resp. ($I = \{vn\}$, $F = \{fn\}$).

Standard methods to measure set similarity may then be used to compare source code. For given SVUPs P and Q we use the F_1 measure[2]:

$$F_1 = 2 \cdot \frac{p \cdot r}{p + r} \quad \text{where} \quad p = \frac{|P \cap Q|}{|P|},\ r = \frac{|P \cap Q|}{|Q|}$$

Fig. 3 shows a dendrogram for an average-linkage clustering using this measure (actually $1 - F_1$ to obtain a distance measure). Codes (a)-(d) correspond to TwoLoopQuickExit, OneLoopQuickExit, OneLoopGuardedIf and Disordered. Additional examples include: Nonsense (similar set of instruction

[1]Note the variable names themselves are not valid identifiers, as the same variable name may occur more than once in the same function, e.g. variable j in code (a) of Fig. 1.

[2]although based on asymmetric precision p and recall r, F_1 itself is symmetric and thus a similarity measure

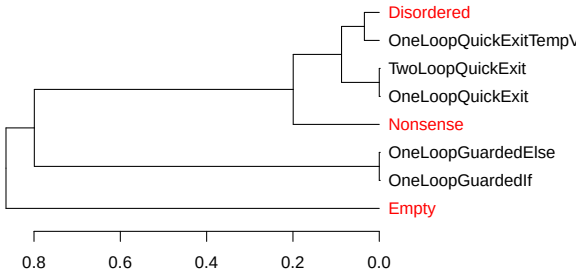


Figure 3: Dendrogram for source codes of Fig. 1 based on F_1-measure on VUP representation.

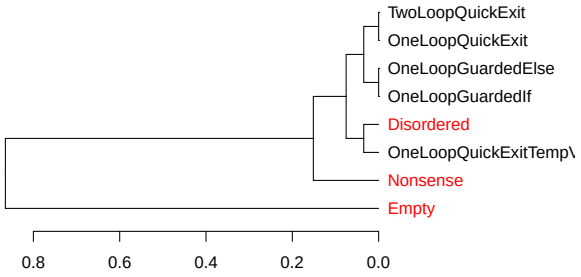


Figure 4: Dendrogram for source codes of Fig. 1 based on F_1-measure on SVUP representation.

Table 1: Example of comparing P and Q (see text).

			d	m(d)
P	1	vn/expr/if/loop/fn	if	(1,6), (2,7)
	2	vn/assign/if/loop/fn	assign	(3,8)
	3	vn/declare/fn	declare	(4,8)
	4	vn/assign/fn		
	5	vn/return/fn		
Q	6	vn/expr/loop/fn	step	un- / assigned
	7	vn/assign/loop/fn	0	1,2,3,4,6,7,8 / 5,9
	8	vn/assign/declare/fn	1	3,4,8 / 1,2,5,6,7,9
	9	vn/return/fn	2	4 / 1,2,3,5,6,7,8,9

set but variables are mixed randomly), OneLoopQuickExitTempVar (the mean value is assigned to a temporal variable before it is returned) and OneLoopGuardedElse (same as OneLoopGuardedIf (Fig. 1(c)) but inverted if-condition). From the dendrogram we learn that codes TwoLoopQuickExit and OneLoopQuickExit become identical as desired. But we can also see that code OneLoopGuardedIf, where a conditional statement encloses the main code, is recognized as very dissimilar. The additional *if*-statement occurs in *all* paths and the simple Jaccard similarity finds this code to be completely different. We address this problem next.

3.3 Reflecting Instruction Embedding

A single conditional statement may introduce an new element in many paths turning them all different (as in codes (a),(c) in Fig. 1). For compensation we could measure a partial similarity between paths (e.g., 80% of path p is contained in q), but a *single instruction* (such as the conditional statement in (c)) would then still weaken *all path similarities*. We therefore propose a different approach in the fashion of edit distance, where we pay a constant cost for a missing path element, which may then be used in many paths.

Given two SVUP representations P, Q, we compare all paths $p \in P, q \in Q$ to identify missing path elements $\delta(p,q) \in I^*$. (For example, δ(*vn/loop/if/fn*,*vn/loop/fn*)=*if*). Many different path combinations may lead to the same $\delta(p,q)$, so by $m(d)$ we denote the set of all pairs $(p,q) \in P \times Q$ with $\delta(p,q) = d$. The matching of paths in P to paths in Q is done iteratively: In the first iteration, we match all paths from P and Q that are identical ($\delta(p,q) = ()$). To match the SVUP representations at minimal cost, in subsequent iterations we identify the missing path element d that unifies the largest number of paths (that is, choose $d = \text{argmax}_x|d(x)|$). Before entering the next iteration, all pairs are removed from $m(\cdot)$ that have been assigned already. We reflect the cost of adding a missing path element by adding it as a *virtual path* to the SVUP P or Q (depending on where it was missing).

Table 1 shows a detailed example. The table on the left shows the paths belonging to SWUP of P and Q. All paths have been numbered for easier reference. The initial map $m(\cdot)$ is shown on the top right; for instance, the path element *if* unifies path #1 of P with #6 of Q, as well as #2 of P with #7 of Q. At the bottom right each line corresponds to an iteration of the matching process. In step 0, paths #5 and #9 are already identical. In the second iteration the path element *if* is chosen from map $m(\cdot)$, because it unifies 2 paths (all others only one). We have thus matched 6 paths in total (step 1 of bottom right table), and only #3, #4, #8 remain unassigned (gray). The map $d(\cdot)$ now offers two alternatives (*assign* and *declare*, both $|m(\cdot)| = 1$), we arbitrarily choose *assign* as the second missing path element for the third iteration. This leaves only path #4 unassigned. The choice of $m(declare)$ has covered paths #4 and #8, so we remove all pairs from $m(\cdot)$ containing any of these (already covered) paths. This ends the matching phase (all $|m(\cdot)| = 0$). We had to add the first missing path element *if* to paths in Q; likewise we had to add the second path element *assign* to P to match #3 against #8. As a penalty for the missing path elements we add them to the respective SVUP, that is, P becomes $\{1,2,3,4,5,assign\}$ and $Q = \{6,7,8,9,if\}$. Taking the established identity of paths into account, this gives us an F_1 value of

$$p = \frac{|P \cap Q|}{|P|} = \frac{4}{6},\ r = \frac{|P \cap Q|}{|Q|} = \frac{4}{5},\ \text{so } F_1 = 2 \cdot \frac{16/30}{44/30} = \frac{8}{11}.$$

Fig. 4 shows the resulting dendrogram. Now OneLoopGuardedIf (Fig. 1(c)) became much more similar to One/TwoLoopQuickExit (Fig. 1(a-b)). But we are still dissatisfied with the high similarities towards Disordered: it uses the same set of embedded instructions (causing the high similarity), but the variable usage is mixed up. The instructions may look identical, but the author did not get the role of variables right and that should degrade the similarity.

3.4 Matching Variables

So we finally revisit the simplification of definition 2. We have hypothesized that similar solution approaches use variables at specific places in the code skeleton. Up to now, we have mixed the usage paths of different variables in the SVUP representation. This will be sorted out next.

Definition 3. Let $\pi_{(v,f)} : \mathcal{P} \to \mathcal{P}, P \mapsto \{p \,|\, p = (v, i_1, \ldots, i_n, f) \in P\}$ be a filtering function that returns only those paths that refer to variable identifier v and function identifier f. From a VUP representation P with a set V of variable identifiers and F of function identifiers, we obtain a **GVUP representation** (grouped VUP) $P' = \{\pi_{(v,f)}(P) \,|\, v \in V, f \in F\}$. That is, a GVUP is a set of VUPs, one VUP set for each variable in each function.

For instance, the code in Fig. 1(b) consists of 4 variables in one function, so the GVUP representation is a set of four VUP representations (one set for each variable). Now, as the GVUP representations P and Q of two source codes are sets of sets, how do we generalize the F_1 calculation? Consider the following example, where we use abbreviated instructions $I = \{a, b, c\}$:

$$P = \{\{x/a/c/f, x/b/c/f\}, \{y/a/f, y/c/f\}, \{z/b/f\}\},$$
$$Q = \{\{x/a/f, x/c/f\}, \{y/b/f\}\}$$

Formally, a direct calculation of $|P \cap Q|$ yields 0 as none of the sets in P is contained in Q. But this does not meet our intention. Note that variable y in P plays the role of x in Q (same for z in P and y in Q). Variables may be renamed without changing the semantics, in fact $|P \cap Q|$ should be 2. The desired semantics is to match variable and function names appropriately, rename them accordingly and calculate the F_1 measure on the obtained $\bigcup_{X \in P} X$ and $\bigcup_{Y \in Q} Y$.

How do we match the sets $X \in P$ against $Y \in Q$? All paths in X or Y refer to the same variable and function name, so we can safely transform the VUP to a SVUP representation and use the measure from section 3.3 to construct a pairwise cost matrix. We employ the Munkres algorithm [8] to find the optimal assignment based on this cost matrix. As the Munkres algorithm performs a 1:1 least-cost assignment, some variables may not get assigned. We match them afterwards in a second pass to the least costly counterpart. (This allows us to match multiple identifiers in one program to the same variable in another, as required when comparing codes (a) and (b) of Fig. 1.)

As an example, for the abovementioned P and Q we would create a cost matrix of all pairwise F_1-values (P-paths in rows, Q-paths in columns):

$$\begin{pmatrix} 0.67 & 0.50 \\ \mathbf{1.00} & 0.50 \\ 0.50 & \mathbf{1.00} \end{pmatrix}$$

The Munkres algorithm optimally assign two of the P-paths to two of the Q-paths (bold face). The third P-path is then associated with the first Q-path, which matches best according to the higher F_1-value. We have thus assigned the variables x and y of P to variable x in Q and may think of renaming all these variables in both codes to, say, u. In the same fashion we may rename the variables of the second assignment to v; reunifying all SVUPs leads us to:

$$P' = \{\ u/a/c/f, u/b/c/f, u/a/f, u/c/f,\ v/b/f\},$$
$$Q' = \{\ u/a/f, u/c/f,\ v/b/f\}$$

The final similarity is obtained by applying the calculations of Sect. 3.3 to both sets P' and Q' (this time with different variable names rather than just generic variable names vn). Although the example did not include different function names, it works in the same way. Fig. 5 shows the resulting dendrogram. As desired, the similarity of code Disordered (Fig. 1(d)) is now worst among all solutions that follow the same solution approach.

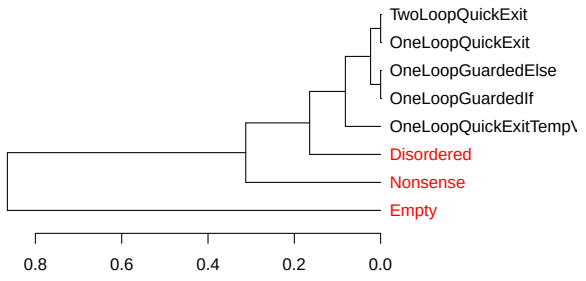


Figure 5: Dendrogram for source codes of Fig. 1 based on F_1-measure on GVUP representation.

4. EXPERIMENTAL EVALUATION

We started to evaluate the proposed approach on real student code submissions and demonstrate its performance on some examples. The submissions were inspected manually and grouped by approach (defining ground truth). While the author of an exercise may have a specific solution in mind, a group of students usually finds multiple ways to solve the exercise. The real data contains nearly identical submissions (potential plagiarism) as well as submissions that appear somewhat chaotic, have superfluous declarations and calculations, or contain artefacts indicating a change in the solution approach over time. Most codes can nevertheless be assigned to solution approaches, but the strong variation in novice's code length misleads edit- and tree-distance such that resulting dendrograms do not match the approaches.

We show results for two exercises, the first exercise asks for the most frequently occurring element in an integer array. To inspire the students to elaborate on different solutions, an additional restriction was given that all values v in the array satisfy $0 \leq v < 10000$. The two main solution approaches were: (1) iterate over all elements in an outer loop, count the frequency of the current element in an inner loop, remember the element that occurs most often; (2) instantiate an array of size 10000 (associating a counter with each possible element in the original array), increment the respective counter while looping over the array and identify the largest entry in the counter array. Apart from these two dominating solutions, three solutions (3) sorted the array first, such that identical values are grouped together, which simplifies frequency counting in a single loop over the array.[3] Finally, (4) there are some exotic solutions which may be considered as a mixture of the discussed solutions. Possibly students became aware of the other approaches by discussing approaches among each others, but had difficulties in solving the task and switched back and forth between them. Usually, elements of all other solutions can be found in them. Fig. 6 shows average-linkage hierarchical clustering

[3]However, when this exercise was handed out, sorting algorithms were not yet discussed (so this solution required background knowledge or a student's own initiative).

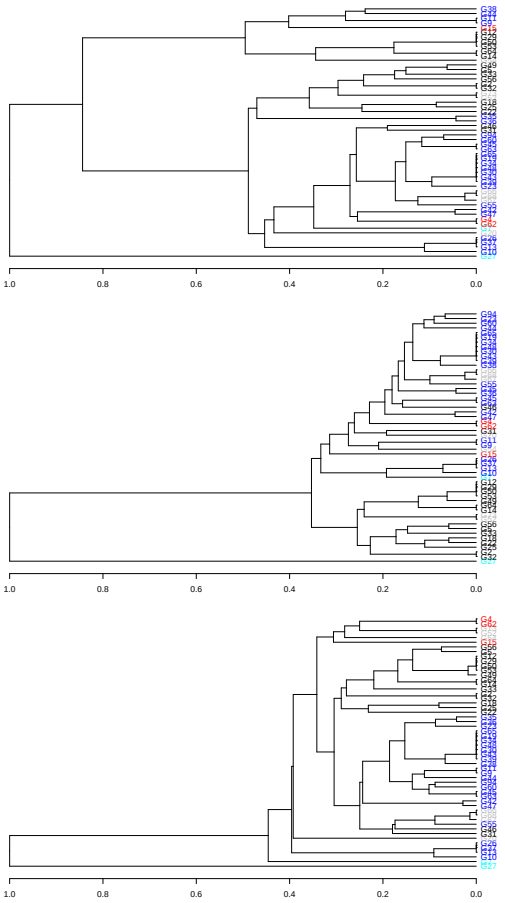


Figure 6: Dendrogram for exercise 'most frequent element' based on F_1-measure (measure from Sect. 3.2, 3.3, and 3.4 from top to bottom).

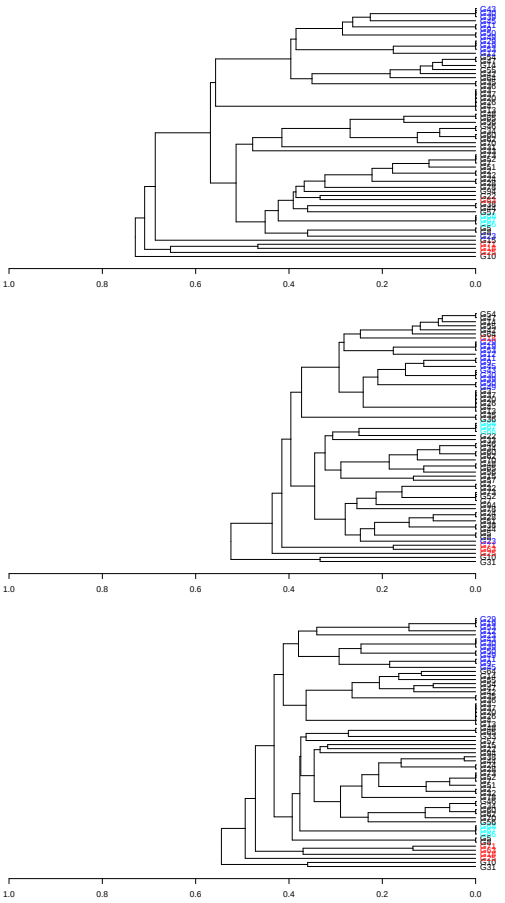


Figure 7: Dendrogram for exercise 'happy number' based on F_1-measure (measure from Sect. 3.2, 3.3, and 3.4 from top to bottom).

results for the similarities from Sect. 3.2, 3.3, and 3.4 from top to bottom. The ground truth is shown by means of the node colors: 1: black, 2: blue, 3: red, 4: gray, incomplete / unfinished: cyan. While the clusters in the top clustering mixes even the two large approaches (1) and (2), the clustering in the middle is much better but does not separate solution (3). This is only achieved by the GVUP-clustering (bottom), which corresponds best to the ground truth. The bubblesort used in (3) uses very similar loops as the main task, so it was crucial to distinguish the role of variables as done in the GVUP-approach.

A second exercise deals with the identification of *happy numbers*[4]. The calculations require the sum of squares of each digit and the submissions differ mainly in the way how this is solved. The intended solution (black) was a loop that successively splits the last digit (using integer division and modulo). Another popular solution is based on string-conversion (blue), a few (somewhat restricted) approaches (red) dealt with different numbers of digits individually (avoiding a loop). Again, the average-linkage clustering for all three proposals is shown in Fig. 7 and the GVUP-approach separates them best. The modulo-approaches (black) subdivide into two major branches, which differ in the way intermediate variables are used. Approaches that use the same kind of utility variables (e.g. to store digits, square of a digit, etc.) are closer matches than approaches that use different sets of utility variables.

[4]`https://en.wikipedia.org/wiki/Happy_number`

5. CONCLUSIONS

Assessing the variety in student's solutions to a programming exercises without having to inspect all codes manually can help a lecturer in many ways. We have proposed a measure that captures how variables and instructions are coupled by means of *variable usage paths*, and use this *fingerprint* to match code from different solutions while at the same time being tolerant to code repetitions. The approach needs to be evaluated further, but the first results appear promising. The nature of the comparison is set-based, which allows us not only to assess similarity (using F_1), but also to use recall and precision. This enables further applications, for instance, we may assess *partial solutions* by the degree how many elements of a *complete solution* they contain (using recall only) or assess the student's degree of programming maturity by investigating the amount of superfluous statements (using precision only).

6. REFERENCES

[1] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier. Clone detection using abstract syntax trees. In *Int. Conf. on Software Maintenance*, pages 368–377. IEEE, 1998.

[2] P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337:217–239, 2005.

[3] J. Broisin and C. Herouard. Design and evaluation of a semantic indicator for automatically supporting programming learning. In *Int. Conf. Educational Data Mining*, pages 270–275, 2019.

[4] M. Chilowicz, E. Duris, and G. Roussel. Syntax tree fingerprinting for source code similarity detection. In *2009 IEEE 17th International Conference on Program Comprehension*, pages 243–247. IEEE, 2009.

[5] Z. Djuric and D. Gasevic. A source code similarity system for plagiarism detection. *The Computer Journal*, 56(1):70–86, 2013.

[6] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113–129, 2010.

[7] O. Karnalim. Syntax trees and information retrieval to improve code similarity detection. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, pages 48–55, 2020.

[8] M. Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[9] C. Ragkhitwetsagul, J. Krinke, and D. Clark. A comparison of code similarity analysers. *Empirical Software Engineering*, 23(4):2464–2519, 2018.

[10] T. Sager, A. Bernstein, M. Pinzger, and C. Kiefer. Detecting similar java classes using tree algorithms. In *Int. Workshop on Mining software repositories*, pages 65–71, 2006.

pyBKT: An Accessible Python Library of Bayesian Knowledge Tracing Models

Anirudhan Badrinath
University of California, Berkeley, CA, USA
abadrinath@berkeley.edu

Frederic Wang
University of California, Berkeley, CA, USA
fredwang@berkeley.edu

Zachary Pardos
University of California, Berkeley, CA, USA
pardos@berkeley.edu

ABSTRACT

Bayesian Knowledge Tracing, a model used for cognitive mastery estimation, has been a hallmark of adaptive learning research and an integral component of deployed intelligent tutoring systems (ITS). In this paper, we provide a brief history of knowledge tracing model research and introduce pyBKT, an accessible and computationally efficient library of model extensions from the literature. The library provides data generation, fitting, prediction, and cross-validation routines, as well as a simple to use data helper interface to ingest typical tutor log dataset formats. We evaluate the runtime with various dataset sizes and compare to past implementations. Additionally, we conduct sanity checks of the model using experiments with simulated data to evaluate the accuracy of its EM parameter learning and use real-world data to validate its predictions, comparing pyBKT's supported model variants with results from the papers in which they were originally introduced. The library is open source and open license for the purpose of making knowledge tracing more accessible to communities of research and practice and to facilitate progress in the field through easier replication of past approaches.

Keywords

Bayesian Knowledge Tracing; Intelligent Tutoring Systems; Educational Software; Python Library

1. INTRODUCTION

Knowledge Tracing [6] has been a well researched approach to estimating students' cognitive mastery in the context of computer tutoring systems [23]. Tutoring systems take a problem-solving, or active approach to learning [2, 1] that often resembles the personalized mastery learning approach researched by Bloom [4]. The model was not originally described using a particular statistical framework; however, the mathematical expressions in the original work are consistent with Bayes Theorem [26], and the canonical model was subsequently coined Bayesian Knowledge Tracing (BKT).

Anirudhan Badrinath, Frederic Wang and Zach Pardos "pyBKT: An Accessible Python Library of Bayesian Knowledge Tracing Models". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 468-474. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

In spite of its growing popularity in the research community, accessible and easy to use implementations of the model and its many variants from the literature have remained elusive. In this paper, we introduce pyBKT[1], a modernized Python-based library, making BKT models and respective adaptive learning research more accessible to the community. The library's interface and underlying data representations are expressive enough to replicate past BKT variants and allow for new models to be proposed. The library is designed with data helpers and model definition functions, allowing for convenient replication and comparison to BKT model variants and subsequently better scientific progression and evaluation of new state-of-the-art knowledge tracing approaches.

The Bayesian Knowledge Tracing model can be described as a Hidden Markov Model (HMM) with observable nodes representing students' known binary problem response sequences obs_t and hidden nodes representing students' latent knowledge state at a particular time step t. Using expectation maximization, pyBKT fits the learn (transmission parameter), and guess, and slip (emission) parameters from historical response logs, with the parameters defined below.

$$\text{prior} = P(L_0)$$

$$\text{learn} = P(T) = P(L_{t+1} = 1 | L_t = 0)$$

$$\text{guess} = P(G) = P(obs_t = 1 | L_t = 0)$$

$$\text{slip} = P(S) = P(obs_t = 0 | L_t = 1)$$

Note that while $P(L_0)$ denotes the prior parameter, we also define $P(L_t)$ as the probability that the student has mastered the skill at time step t. Bayesian Knowledge Tracing updates $P(L_t)$ given an observed correct or incorrect response to calculate the posterior with:

$$P(L_t | obs_t = 1) = \frac{P(L_t)(1 - P(S))}{P(L_t)(1 - P(S)) + (1 - P(L_t))P(G)}$$

$$P(L_t | obs_t = 0) = \frac{P(L_t)P(S)}{P(L_t)P(S) + (1 - P(L_t))(1 - P(G))}$$

The updated prior for the following time step, which incorporates the probability of learning from immediate feedback and any other instructional support, is defined by:

$$P(L_{t+1}) = P(L_t | obs_t) + (1 - P(L_t | obs_t))P(T)$$

The standard BKT model assumes no forgetting:

$$P(F) = P(L_{t+1} = 0 | L_t = 1) = 0$$

[1] `https://github.com/CAHLR/pyBKT`

2. RELATED WORK

Since the introduction of the standard BKT model by Corbett and Anderson [6], many variants of BKT have been proposed which multiplex, or condition the four parameters of prior, learn, guess, and slip on different factors such as question type or student. KT-IDEM (Item Difficulty Effect) [20] captured item performance variance within skill by allowing different guess and slip values to be fit per item. Baker et al. [3] proposed a hybrid model using regression to help determine if a student's response was a guess or a slip based on context.

Other modifications focused on conditioning the learn rate and prior knowledge parameter [19]. Yudelson et al. [30] explored student-level parameter individualization, finding that the learn rate provided better predictive performance than prior when individualized to each student. Learn rate has also been conditioned based on the item [17], educational resource (e.g., a video in an online course) [21], or type of instructional intervention seen by the student at each opportunity [13] or order in which items or resources were seen [18]. The assumption of no forgetting was relaxed in Qiu et al. [25], finding that response correctness decreased based on time elapsed since the last response. This decrease was better modeled in BKT with conditional forget rates than with increased slip rates. González-Brenes et al. [9] created a hybrid BKT model that factored in a variety of features, including response time. An overview of other variations of BKT and logistic approaches to learner response prediction can be found in Pelánek [23].

Neural network approaches to knowledge tracing have gained in momentum since the introduction of Deep Knowledge Tracing (DKT; [24]). Some have explored the reasons for DKT's apparent accuracy improvement as compared to BKT and have attributed its success to its high dimensional hidden space and ability to observe interleaved skills in a single model [14]. However, most papers using neural approaches compare only to the standard BKT model proposed in the 1990s and not to the more modern variants. A noteworthy result of caution was reported in Khajah et al. [11], in which it was found that simply enabling the forget parameter of standard BKT led to performance on par with DKT on several datasets.

There has been a brief history of BKT implementation frameworks. Several BKT variants have used Kevin Murphy's Bayes Net Toolbox (BNT) for MATLAB [15], with a subsequent wrapper for that toolbox releases catering to knowledge tracing [5]. Yudelson et al. [30] produced a C++ implementation[2] of BKT with a command line interface that included support for individualized parameters. Finally, Xu et al. [29] created a C++ implementation of BKT with a MATLAB interface and support for parallelization and conditioning of all parameters based on both problems and passive resources (e.g., a learning rate for a video).

3. PYBKT LIBRARY DETAILS

The pyBKT library builds off of xBKT developed by Xu et al. [29] and is released under an MIT license. The library is compatible with all platforms (Linux, Windows, Mac OS), primarily utilizing NumPy [28] for computation. It is available on the Python PyPi repository, with installation accomplished through a pip one-liner: `pip install pyBKT`. To increase performance, we additionally supply routines which utilize C++ libraries and Eigen/LAPACK, which is an optimized linear algebra package with OpenMP support. This accelerated version requires a C++ compiler and is currently only tested on Linux and macOS.

We created a Scikit-learn style [22] Model class abstraction and accompanying data helpers that further facilitate the accessibility and expressive power of pyBKT. With one-line fit, predict, evaluate, parameter initialization, and cross-validate methods, pyBKT offers ease of use in ingesting response data and applying BKT models and supported variants. We explore the interface to these methods in the next subsection. We then detail the internal data structures, computations, and motivations behind the development of the two implementations of pyBKT, in pure Python and the accelerated C++/Python, along with runtime evaluation.

3.1 Interface

pyBKT's interface is modeled after Scikit-learn's accessible frontend interface for machine learning models [22]. The ease-of-use in the pyBKT Model class abstraction allows for increasingly expressive BKT code without the usability sacrifices of past BKT libraries. We aim for the library to be easy to learn for beginners while still useful for experienced users conducting knowledge tracing research. Further, it provides a gateway into exploring multiple model extensions from the literature, which have been shown to be capable competitors to DKT [11] and able to address inequities in unmodeled differences in learning and prior ability between students [7]. Supported BKT extensions include: KT-IDEM [20], KT-PPS (Prior Per Student) [19], BKT+Forget [11], Item Order Effect [18] and Item Learning Effect [17, 21]. These model extensions are referred to as `multigs`, `multiprior`, `forgets`, `multipair`, and `multilearn`, respectively, in the model interface.

The Model class abstraction supports creating, fitting, predicting, cross-validating and evaluating BKT models using any combination of supported extensions. Additional features include specifying model parameter initialization before fitting, custom cross-validation fold assignment, and multiple accuracy and error metrics - including support for generic user-defined or Scikit-learn imported metrics. Common dataset formats are made easier to ingest through automatic detection of familiar column headers seen in Cognitive Tutor [12] and ASSISTments datasets [10]. Defaults for all customizable parameters such as random seed, parallelization, model variants, and evaluation metric(s) are provided when they are not specified.

We demonstrate a few of the library's basic capabilities in parameter initialization, fitting, and parameter output in the below code snippet using the learned parameters of the "Polynomial Factors" skill from the 2009-2010 ASSISTments dataset[3]. Note that all parameters of all skills found in the dataset are fit unless otherwise specified.

[2] https://github.com/myudelson/hmm-scalable

[3] `https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data`

```
>>> from pyBKT.models import Model
>>> model = Model()
>>> model.fit(data_path = 'assistments.csv')
>>> model.params().loc[('Polynomial Factors')]
param   class      value
prior   default  0.17452
learns  default  0.13378
guesses default  0.25502
# ...
```

The internal data helper functionality converts any response-per-row comma or tab separated file into the internal pyBKT data format. It is designed to convert input columns using default column mappings for skill name, student identifiers, correctness, etc. automatically for Cognitive Tutor/ASSISTments data and configurable with one line for any other dataset. It provides increased flexibility to the user, allowing for consistency across fit and predict/evaluate phases.

The included evaluation metrics are root-mean squared error (RMSE), accuracy with a threshold of 0.5, and area under the ROC curve (AUC). Custom metrics in the format of two-parameter Python functions are supported for evaluation and cross-validation, such as the regression or classification metrics from `sklearn.metrics` or `keras.metrics`.

The cross-validate function provides a one-line interface for fitting and evaluating any combination of model variants with one or more error metrics. For the following example, we specify a particular column to use for the multilearn (answer type) along with a multigs model trained on the default template ID for ASSISTments. A skill or combination of skills can be specified along with the seed and number of folds (optional).

```
>>> model = Model(seed = 42, parallel = True)
>>> model.crossvalidate(data_path = 'assistments.csv',
      skills = ['Circle Graph', 'Box and Whisker'],
      multigs = True, multilearn = 'answer_type', metric =
      ['auc', sklearn.metrics.mean_absolute_error])
skill                mean_absolute_error      auc
Circle Graph                     0.41565 0.72782
Box and Whisker                  0.33906 0.67991
```

3.2 Internal Implementation Details

The pure Python and C++/Python implementations of pyBKT both make use of optimized programmatic methods, efficient internal data and model representation, multithreaded model fitting, and optimized linear algebra libraries. The model fitting consists of a typical Expectation Maximization (EM) function for a Hidden Markov Model performing forward and backward passes over the sequential data to continuously update BKT parameters. We implement these passes using a parallelized iterative dynamic programming approach on the input data.

3.2.1 Model Representation

In the context of model variants with multiple learn, guess, slip, or forget rates, a subscript $P(T_i)$, $P(G_i)$, $P(S_i)$, $P(F_i)$ denotes the corresponding probability, or rate, for class $i = 1...m$, respectively.

Initial and fit BKT model parameters are represented using a Python dictionary. Inside of this dictionary, we store A, which is a collection of matrices with each 2x2 matrix corresponding to the learning and forgetting probabilities for each learn class in order to aid in efficient matrix multiplication during fitting. A has the format where m is the total number of learn rates:

$$\left[\begin{bmatrix} P(\neg T_1) & P(F_1) \\ P(T_1) & P(\neg F_1) \end{bmatrix}, \cdots, \begin{bmatrix} P(\neg T_m) & P(F_m) \\ P(T_m) & P(\neg F_m) \end{bmatrix}\right]$$

We define α as a set of 2-length vectors each corresponding to $[P(\neg L_t), P(L_t)]$ for all time steps for a specific student. Similarly, π_0 stores information about the prior, in the format of $P(\neg L_0), P(L_0)$.

3.2.2 EM and BKT Algorithm

We use Expectation Maximization to fit model parameters, shown to provide desirable convergence properties, given plausible initial parameter values [16]. Inside the EM and inference algorithms, we use several intermediate data structures and vectorization to improve computational efficiency in fitting the models. To calculate $\alpha[t+1]$ given $\alpha[t]$, we multiply it by the part of the learn/forget transition matrix A corresponding to the learn class of time step t. We element-wise multiply by the vector likelihoods which consists of $[P(G), P(\neg S)]$ or $[P(\neg G), P(S)]$ for the corresponding guess class of time step t, depending on whether the student answers correctly or not, respectively. Finally, normalizing this vector results in $\alpha[t+1]$. We demonstrate the algorithm for an example iteration of the BKT algorithm with learn class 1, guess class 1, and an incorrect response observed ($obs = 0$) at time step t.

$$\begin{bmatrix} P(\neg L_{t+1}) \\ P(L_{t+1}) \end{bmatrix} = \begin{bmatrix} P(\neg T_1) & P(F_1) \\ P(T_1) & P(\neg F_1) \end{bmatrix} \left(\begin{bmatrix} P(\neg L_t) \\ P(L_t) \end{bmatrix} \circ \begin{bmatrix} P(\neg G_1) \\ P(S_1) \end{bmatrix} \right)$$

$$= \begin{bmatrix} P(\neg T_1) * P(\neg L_t | obs_t) + P(F_1) * P(L_t | obs_t) \\ P(T_1) * P(\neg L_t | obs_t) + P(\neg F_1) * P(L_t | obs_t) \end{bmatrix}$$

At the end of this α calculation, we perform the E-step of the EM algorithm by recursively calculating an expectation γ for each time step by backtracking through the learned latent states. We can then take the global average of the expectations of the learn/forget transition matrices, guess/slip vectors, and priors during the M-step and use these as the parameters for the next iteration of EM. In terms of the number of students S and the typical sequence length for each student T, the model fitting algorithm's asymptotic time complexity for standard BKT is $\Theta(TS)$.

3.2.3 C++/Python Implementation Details

We use a C++ extension to perform the EM iterative updates and matrix multiplication for the model fit and prediction process. This allows us to use efficient linear algebra libraries in C++[4] and benefit from greater support for multithreading through OpenMP.

We use Eigen to perform the matrix operations. There are many technical advantages of using Eigen with a linear al-

[4]Boost was previously used as a connector between Python and the C++ extension, but it has been deprecated since pyBKT 1.2.2, resulting in a 3-5x performance increase.

gebra heavy model such as pyBKT. Eigen provides efficient, thread-safe matrices and arrays, while being a relatively portable package distributed along with pyBKT. It allows for lazy evaluation, expression templates and compiler optimizations.

We use OpenMP for parallelizing the demanding model fitting process. OpenMP is a universally accepted multithreading library for C/C++ that exploits multicore processors with low overhead. With a shared memory space for forked threads, OpenMP avoids overhead for inter-process communication (IPC) unlike Python's multiprocessing. With Eigen's explicit support for OpenMP-based multithreading, heavy matrix operations and iterative processes are further optimized in pyBKT.

3.2.4 *Pure Python Implementation Details*

We wish to maintain the accessibility of pyBKT across all platforms while maintaining as much efficiency as possible. To do this, a pure Python implementation, without C++ extensions, is included. This implementation provides quick access to any user, including on Windows, wishing to run a BKT model without the hassles of compilation and complex dependencies. We relax this version of the library's requirements to include mostly native modules along with the widely supported NumPy.

NumPy is used for the matrix operations in the pure Python build of pyBKT as it is the most efficient and widely used numeric computational library in Python. Technically, it provides impressive single-threaded performance. Similarly to other optimized mathematical libraries such as Eigen, NumPy employs code vectorization, efficient memory mapping techniques for sparse matrices, and compiler optimizations.

Since NumPy is primarily a single-threaded application with little support for multicore scaling, we use Python's native multiprocessing library for parallelizing the model fitting. It provides native CPython support for multicore scaling to bypass the Global Interpreter Lock (GIL), which allows only one running thread of execution within a process. Although different Python implementations (i.e JPython) exist to disable the GIL or remove the memory overhead, we use a native module for simplicity and speed.

3.3 Runtime Evaluation

We compare the runtime performance of the pure Python and C++/Python implementations of pyBKT on five typical model fitting and prediction tasks. We present two sets of tasks, fitting and prediction on synthetic data, that additionally showcases the way in which the runtimes scale with the size of input data for both implementations of pyBKT. Each of the tasks are averaged over several runs for both implementations of pyBKT on a machine with 2 x Intel(R) Xeon(R) CPU E5-2620 v3 CPUs at 2.4Ghz with 256GB of system RAM. The results are shown in Table 1.

We evaluate the runtimes using two metrics. The scaling factor is defined as the ratio of the runtime of the larger input and the smaller input for a set of prediction or fitting tasks. The speedup is defined as the ratio of the runtime of our C++/Python implementation to our pure Python implementation.

The first four tasks perform synthetic data generation for 500 students and 5,000 students respectively with a sequence length fixed at 100 followed by prediction or fitting. These tasks illustrate a typical medium and large workload for model fitting and prediction tasks. The generated synthetic data is fit or predicted using a standard BKT model. It is clear that as the number of students scales, the pure Python implementation of pyBKT performs and scales more poorly with the number of students. The C++/Python implementation shows a nearly 150-600x speedup for fitting and 15-30x speedup for prediction as compared to the pure Python version. In comparison to its predecessor xBKT (MATLAB), the C++/Python version of pyBKT gains a 3-4x speedup across all fitting and prediction tasks. In Xu et al. [29], it is noted that xBKT outperforms BNT by 10,000x, which suggests a 30,000-40,000x speedup of pyBKT as compared to BNT.

The final task performs a cross-validated prediction task for a selected skill in the Cognitive Tutor dataset. We use a variety of models (standard, multigs, multiprior) to test predictive accuracy and measure its runtime. This task is around 65x slower in the pure Python implementation.

While the runtimes and the overall scaling of the pure Python port with respect to the size of input data are significantly poorer for each task, that is an expected trade-off with regards to accessibility and portability. For an end-user that is training and testing moderately-sized BKT models or evaluating models, they would benefit from a portable and universal BKT model which can handle a moderate input size with relative efficiency. For heavier research-oriented or production-oriented tasks, the C++/Python implementation is recommended since it generally performs much more efficiently.

4. DATA SUFFICIENCY ANALYSIS

We examine the data sufficiency requirements of the standard BKT model by exploring trade-offs between input size and parameter error and mastery estimation accuracy. We define the input size as the magnitude of the number of students and the average sequence length. Through the first analysis, practitioners may gain an intuition for the minimum cohort size and minimum number of questions answered per student per skill to effectively apply BKT. Our second analysis in this section focuses on mastery estimation accuracy, also using synthetic data. This analysis depicts how the worst-case expected mastery estimation accuracy decreases as a function of sequence length for a given set of prior, guess, and slip parameters.

For the following analyses, we generate synthetic data, both responses and mastery states, from pyBKT using ground truth parameter values set to common values seen for Algebra skills[5]. In doing so, we are able to calculate the error of the fit parameters and accuracy of the mastery estimation.

[5]prior=0.08, guess=0.15, slip=0.05, forget=0 and learn=0.3 for the first analysis and variable for the second

Table 1: Comparison of runtimes, scale factor and speedup between Python and C++/Python implementations of pyBKT.

Test Description	Pure Python		C++/Python		Speedup
	Runtime	Scaling	Runtime	Scaling	
Synthetic Data Generation, Model Fit (500 students)	160.12s		1.07s		149.64x
Synthetic Data Generation, Model Fit (5,000 students)	1,596.30s	9.97x	2.62s	2.45x	609.27x
Synthetic Data Generation, Model Predict (500 students)	8.02s		0.50s		16.04x
Synthetic Data Generation, Model Predict (5,000 students)	67.08s	8.36x	2.42s	4.84x	27.71x
Cross-validation, Cognitive Tutor	320.28s	-	4.79s	-	66.86x

4.1 Synthetic Model Fit Accuracy

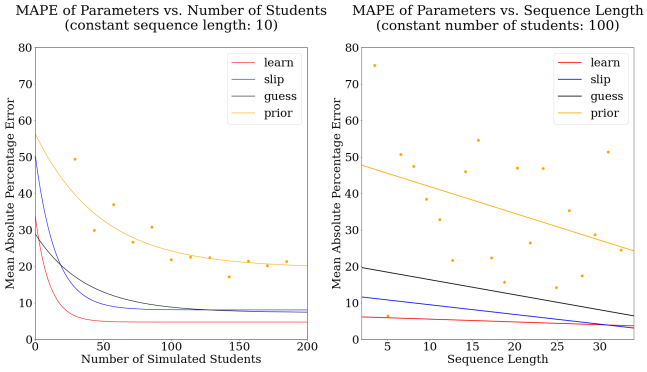
The synthetic generation of data is performed for input sizes from 10 to 200 students for all sequence lengths from 2 to 35. While 200 is not an uncommon number of students to have in a cohort, more than 30 responses to a single skill would be unusual for a student. Each combination of input size is averaged over five fits, each of which includes the best model over 20 random EM fit initializations.

The error of the model's fit parameters are analyzed using the mean absolute percentage error (MAPE) in relation to predefined ground truth generating parameters. We plot the fitting error of the learn, slip, guess, and prior parameters as a function of the number of synthesized students (Figure 1, left) and length of synthesized response sequences for each student (Figure 1, right). While all data points cannot be visualized on a single plot, we show the data points for the prior parameter as an example.

For all parameters, there is a clear negative and exponential error decay with respect to the number of students. This is consistent with an expected asymptotic behaviour when increasing the number of students in the fitting procedure. Learn and slip parameters asymptote at around 50 students while guess and prior do so after 100, given a sequence length of 10.

There seems to be a slowly decreasing linear relationship between the typical sequence length and parameter fit error, with the prior parameter showing the greatest improvement in MAPE. These analyses show that there is not nearly as much benefit to fitting error by increasing the sequence length (i.e., giving students more problems) as there is by increasing the number of students.

Figure 1: Mean absolute percentage error (MAPE) of fit parameters as a function of number of students (left) and sequence length (right).



4.2 Synthetic Mastery Estimation Accuracy

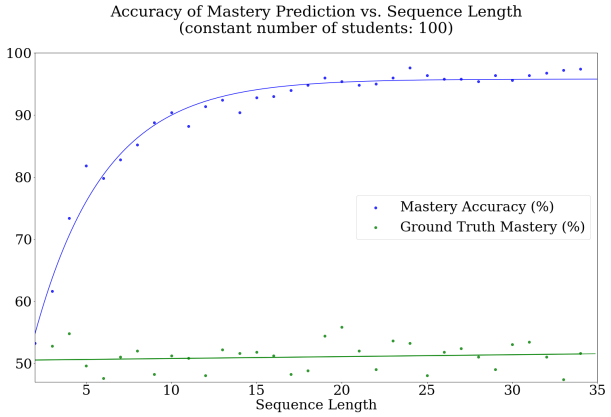
We look to evaluate the worst-case accuracy of the standard BKT model's mastery estimation, its most common task within a computer tutoring system, using a mastery threshold of $P(L_t) \geq 0.95$ on simulated problem solving sequences. We use the same predetermined ground truth BKT parameters and data generation methodology as in the previous subsection with the exception of the learn rate, which is set dynamically in this analysis.

It is known that the probability of mastery will converge to $1 - forgets$ in a finite number of time steps given a learn rate > 0 [27]. This means that cognitive mastery estimation accuracy will increase with respect to sequence length, as student mastery state becomes more homogeneous.

In order to model the worst-case mastery estimation of the model, we find the learn rate that will produce an average probability of mastery close to 0.5 across students at the final time step τ for all our chosen sequence lengths, thus preventing a trivial estimation of a majority mastery state. We find the learn rate via grid search with a granularity of 0.001 since this cannot be solved analytically [27].

The mastery estimation accuracy exponentially decays upward with respect to sequence length with a Pearson correlation $R_{\log(acc)} = 0.73$ as shown in Figure 2. The mastery estimation accuracy in our analysis can be observed to asymptote around a sequence length of 15. This suggests that the worst-case mastery estimation accuracy scenario can be mitigated, given our chosen predefined parameter values, with an average response sequence length per skill of 15 or greater.

Figure 2: Worst-case accuracy of mastery estimation as a function of sequence length for the standard BKT model.



5. EVALUATION OF PYBKT MODELS

To gauge the validity of the BKT models and extensions supported by pyBKT, we perform predictive model evaluations replicating those found in prior literature. Models were evaluated by performing five-fold cross-validation on the ASSISTments 2009-2010 Skill Builder dataset, the Cognitive Tutor 2006-2007 Bridge to Algebra dataset[6], and specialized datasets from Feng et al. [8] and Piech et al. [24] in order to validate more specific model extensions. While there are slight differences between our results and the results of other papers, we believe this is due to our random parameter initialization and small differences in fitting methods. Our complete model evaluations, data, and results are located in a pyBKT examples repository[7].

5.1 Standard BKT and BKT+Forget

In general, the predictive accuracy of the models generated by pyBKT are in line with what others have observed from BKT. For instance, in Khajah et al. [11], they found that by fitting the basic BKT model on the train/test split provided by Piech et al. [24] from the 2009-2010 ASSISTments dataset for each skill, they obtained an AUC of 0.73, whereas pyBKT predicts with an AUC of 0.76. Similarly, when Khajah et al. [11] added the forgets parameter into their model, they achieved an AUC of 0.83, exactly the same AUC we achieve using pyBKT.

5.2 KT-IDEM

When using the KT-IDEM model on the ASSISTments 2009-2010 Skill Builder dataset, pyBKT achieves an average AUC increase of 0.01932, which is very close to the 0.021 average increase reported in Pardos and Heffernan [20]. While the exact subset of data was not exactly specified in that paper, we used the ten skills with the most responses, using different template ids as the guess/slip classes as prescribed in [20]. Since the ASSISTments data has a very high average response to template ratio ($\sim$1,000), the KT-IDEM model performs very well compared to the standard BKT model using RMSE as the metric of comparison, being lower or equal in nine of the ten skills selected.

5.3 KT-PPS

The Prior Per Student model was applied to the ASSISTments' Groups of Learning Opportunities dataset [8] consisting of 42 problem sets. In Pardos and Heffernan [19], it was found that the KT-PPS model performed better than the standard BKT model on 30 out of the 42 problems sets, as evaluated on the predictions of the last response of each student's response sequence. This was achieved using a variant of KT-PPS that models a high and low prior and assigns students to the high prior if they answer correctly on the first problem of the problem set, and to the low prior otherwise. The high prior was set to an ad-hoc value of 0.90 and the low prior to 0.15 in that work.

The pyBKT replication of this model is done without true multiple prior modeling. Instead, when the multiprior option is set to True, $P(L_0)$ is set to 0 and a dummy time step is created at the beginning of the sequence. Three learn rates are created, the first corresponding to the high prior, the second to the low prior, and the third corresponding to the standard $P(T)$ applied between all subsequent time steps. The initial values of these virtual priors were set to the ad-hoc values from [19]; however, since pyBKT does not support parameter fixing as of this writing, these parameters were learnable. With these settings, pyBKT's KT-PPS performs better than standard BKT on 27 out of 42 of the problem sets. The small difference in prediction accuracy of this model may be attributable to the difference in the algorithm regarding fixed parameters, but the similarity in performance is promising.

[6] `https://pslcdatashop.web.cmu.edu/KDDCup`

[7] `https://github.com/CAHLR/pyBKT-examples`

5.4 Item Order and Item Learning Effect

Results from the Item Order Effect [18] and Item Learning Effect [17] papers were not focused on response prediction improvement. In fact, no prediction accuracy results were provided. Instead, the purpose of the models was to compare the learn rates of classes to flag effective and ineffective items and orders. The examples repo of pyBKT depicts such differences. Nevertheless, a modest 0.01 RMSE improvement for both model variants was obtained compared to the standard BKT model.

6. CONCLUSIONS

We demonstrated pyBKT as a seamlessly installable, efficient, and portable Python library with model extensions such as KT-IDEM, KT-PPS, BKT+Forgets, Item Order Effect and Item Learning Effect. The Model class abstraction in pyBKT provides an expressive way to interact with the BKT model extensions with ease, with one-line methods to create, initialize, fit, predict, evaluate, and cross-validate any combination of BKT model extensions. We measured the runtime of pyBKT to be nearly 3x-4x faster than its predecessor, xBKT, and nearly 30,000x faster than BNT, a standard BKT implementation. Through the analyses presented, we established 50 as a reasonable number of students to achieve convergence to canonical parameter values with any average student sequence length and 15 as a reasonable sequence length to mitigate worst-case mastery estimation accuracy. Lastly, through real-world dataset analyses, we showed the validity of the model implementation through its agreement with past results using established software.

Acknowledgments

We recognize Matthew J. Johnson, who co-developed xBKT, the predecessor to pyBKT, and Cristian Garay, who developed the initial Python and Boost adaptation of xBKT.

References

[1] Vincent AWMM Aleven and Kenneth R Koedinger. An effective metacognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive science*, 26(2):147–179, 2002.

[2] Yuichiro Anzai and Herbert A Simon. The theory of learning by doing. *Psychological review*, 86(2):124, 1979.

[3] Ryan SJ Baker, Albert T Corbett, and Vincent Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International conference on intelligent tutoring systems*, pages 406–415. Springer, 2008.

[4] Benjamin S Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13(6):4–16, 1984.

[5] Kai-min Chang, Joseph Beck, Jack Mostow, and Albert Corbett. A bayes net toolkit for student modeling in intelligent tutoring systems. In *International Conference on Intelligent Tutoring Systems*, pages 104–113. Springer, 2006.

[6] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4): 253–278, 1994.

[7] Shayan Doroudi and Emma Brunskill. Fairer but not fair enough on the equitability of knowledge tracing. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, pages 335–339, 2019.

[8] Mingyu Feng, Neil T Heffernan, Joseph E Beck, and Kenneth R Koedinger. Can we predict which groups of questions students will learn from?. In *EDM*, pages 218–225, 2008.

[9] José González-Brenes, Yun Huang, and Peter Brusilovsky. General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In *EDM*, pages 84–91. University of Pittsburgh, 2014.

[10] Neil T Heffernan and Cristina Lindquist Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4): 470–497, 2014.

[11] Mohammad Khajah, Robert V Lindsey, and Michael C Mozer. How deep is knowledge tracing? *arXiv preprint arXiv:1604.02416*, 2016.

[12] Kenneth R Koedinger, John R Anderson, William H Hadley, and Mary A Mark. Intelligent tutoring goes to school in the big city. 1997.

[13] Chen Lin and Min Chi. Intervention-bkt: incorporating instructional interventions into bayesian knowledge tracing. In *International conference on intelligent tutoring systems*, pages 208–218. Springer, 2016.

[14] Shirly Montero, Akshit Arora, Sean Kelly, Brent Milne, and Michael Mozer. Does deep knowledge tracing model interactions among skills? In *EDM*, 2018.

[15] Kevin Murphy et al. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.

[16] Zachary Pardos and Neil Heffernan. Navigating the parameter space of bayesian knowledge tracing models: Visualizations of the convergence of the expectation maximization algorithm. In *EDM*, pages 161–170, 2010.

[17] Zachary A Pardos and Neil T Heffernan. Detecting the learning value of items in a randomized problem set. In *AIED*, pages 499–506, 2009.

[18] Zachary A. Pardos and Neil T. Heffernan. Determining the significance of item order in randomized problem sets. In *EDM*, pages 111–120, 2009.

[19] Zachary A Pardos and Neil T Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 255–266. Springer, 2010.

[20] Zachary A Pardos and Neil T Heffernan. Kt-idem: Introducing item difficulty to the knowledge tracing model. In *International conference on user modeling, adaptation, and personalization*, pages 243–254. Springer, 2011.

[21] Zachary A Pardos, Yoav Bergner, Daniel T Seaton, and David E Pritchard. Adapting bayesian knowledge tracing to a massive open online course in edx. *EDM*, 13: 137–144, 2013.

[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12: 2825–2830, 2011.

[23] Radek Pelánek. Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*, 27(3-5):313–350, 2017.

[24] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Advances in neural information processing systems*, pages 505–513, 2015.

[25] Yumeng Qiu, Yingmei Qi, Hanyuan Lu, Zachary A Pardos, and Neil T Heffernan. Does time matter? modeling the effect of time with bayesian knowledge tracing. In *EDM*, pages 139–148, 2011.

[26] Jim Reye. Student modelling based on belief networks. *International Journal of Artificial Intelligence in Education*, 14(1):63–96, 2004.

[27] Brett van De Sande. Properties of the bayesian knowledge tracing model. *EDM*, 5(2):1–10, 2013.

[28] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[29] Y Xu, M J Johnson, and Z A Pardos. Scaling cognitive modeling to massive open environments. In *Proceedings of the Workshop on Machine Learning for Education at the 32nd International Conference on Machine Learning (ICML)*, 2015.

[30] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*, pages 171–180. Springer, 2013.

On the Limitations of Human-Computer Agreement in Automated Essay Scoring

Afrizal Doewes, Mykola Pechenizkiy
Eindhoven University of Technology
{a.doewes, m.pechenizkiy}@tue.nl

ABSTRACT

Scoring essays is generally an exhausting and time-consuming task for teachers. Automated Essay Scoring (AES) facilitates the scoring process to be faster and more consistent. The most logical way to assess the performance of an automated scorer is by measuring the score agreement with the human raters. However, we provide empirical evidence that a well-performing essay scorer from the quantitative evaluation point of view are still too risky to be deployed. We propose several input scenarios to evaluate the reliability and the validity of the system, such as off-topic essays, gibberish, and paraphrased answers. We demonstrate that automated scoring models with high human-computer agreement fail to perform well on two out of three test scenarios. We also discuss the strategies to improve the performance of the system.

Keywords

Automated Essay Scoring, Testing Scenarios, Reliability and Validity

1. INTRODUCTION

Automated Essay Scoring (AES) system is a computer software designed as a tool to facilitate the evaluation of student essays. Theoretically, AES systems work faster, reduce cost in term of evaluator's time, and eliminate concerns about rater consistency. The most logical way to assess the performance of an automated scorer is by measuring the score agreement with the human raters. The score agreement rate must exceed a specific threshold value to be considered as having a good performance. Consequently, most studies have focused on increasing the level of agreement between human and computer scoring. However, the process of establishing reliability should not stop with the calculation of inter-coder reliability, because automated scoring poses some distinctive validity challenges such as the potential to misrepresent the construct of interest, vulnerability to cheating, impact on examinee behavior, and users' interpretation on score and use of scores [1]. Bennet and Bejar [2] have argued that reliability scores are limited in their reliance on human ratings for evaluating the performance of automated scoring primarily because human graders are fallible. Humans raters may experience fatigue and have problems with scoring consistency across time. Reliability calculations alone are therefore not adequate as the current trend for establishing validity [3]. A well-performing essay scorer from the quantitative evaluation perspective is too risky to be deployed before evaluating the system's reliability and validity.

The initial attempt to discuss validity issues regarding automated scoring in a larger context of a validity argument for the assessment was made by Clauser et al. [4]. They presented several outlines of the potential validity threats that automated scoring would introduce to the overall interpretation and use. Enright and Quinlan [5] discussed how the evidence for a scoring process that uses both human and e-rater scoring is relevant to validity argument. They described an e-rater model which was proposed to score one of the two writing tasks on the TOEFL-iBT writing section. Automated scorer was investigated as a tool to complement to human judgement on essays written by English language learners.

Several criticisms for Automated Essay Scoring (AES) system were highlighted in [6]. They argued that there were limited studies on how effective automated writing evaluation was used in writing classes as a pedagogical tool. In their study, the students gave negative reactions towards the automated assessment. One of the problems was that the scoring system favored lengthiness; higher scores were awarded to longer essays. It also overemphasized the use of transition words, which increased the score of an essay immediately. Moreover, it ignored coherence and content development as an essay could achieve a high score by having four or five paragraphs with relevant keywords, although it had major coherence problems and illogical ideas. Another concern is described in [7]. Specifically, knowing how the system evaluates an essay may be a reason why students can fool the system into assigning a higher score than what is warranted. They concluded that the system was not ready yet as the only scorer, especially for high-stakes testing, without the help of expert human raters.

Most researchers agree that human - automated score agreement still serves as the standard baseline for measuring the quality of machine score prediction. However, there is an inherent limitation with this measurement because the agreement rate is usually derived only from the data used for training and testing the machine learning model. The aim of this paper is to highlight some limitations of standard performance metrics used to evaluate automated essay scoring model, using several input scenarios to evaluate the reliability and the validity of the system, such as off-topic essays, gibberish, and paraphrased answers. We show empirical evidence that a well-performing automated essay scorer, with high agreement rate between human-machine, is not necessarily ready for deployment for operational use, since it fails to perform well on two out of three test scenarios. In addition, we also discuss some strategies to improve the performance of the system. This paper begins with the explanation of the quantitative performance acceptance criteria for an automated scoring model from [1]. Then, we present the experiment settings, including the training algorithm and the essay features, for creating the model. Afterwards, we discuss the experiment results, model performance analysis, reliability and validity evaluation and the strategies for improvement, and finally, we conclude our work.

Afrizal Doewes and Mykola Pechenizkiy "On the Limitations of Human-Computer Agreement in Automated Essay Scoring". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 475-480. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

2. SCORE AGREEMENT EVALUATION

According to Williamson et al. in [1], the following are some of the acceptance criteria used for evaluation of automated scoring with respect to human scores when automated scoring is intended to be used in conjunction with human scoring:

1. Agreement of scores between human raters and computer

Agreement between human scores and automated scores has been a long-established measure of the performance of automated scoring. This is to measure whether the agreement satisfies a predefined threshold. The quadratic weighted kappa (QWK) between automated and human scoring must be at least .70 (rounded normally). It is important to note that the performance of automated scorer will rely on the quality of the human scoring. Therefore, the interrater agreement among human raters must first be reliable.

2. Degradation from human-human score agreement

The human-automated scoring agreement cannot be more than .10 lower than the human-human agreement. This standard prevents the case in which automated essay scoring may reach the .70 threshold but still be notably deficient in comparison with human scoring. In addition, it does not rule out the cases in which the automated–human agreement has been slightly less than the .70 threshold, but very close to a borderline performance for human scoring. For example, a human-computer QWK of .69 and human-human QWK of .71. Such model is approved for operational use based on being highly similar to human scoring. Moreover, Williamson et al. stated that it is relatively usual to observe automated-human agreements that are higher than the human-human agreements for tasks that predominantly target linguistic writing quality (e.g. GRE Issue and TOEFL Independent tasks).

3. Standardized mean score difference between human and automated scores.

Another measure for association of automated scores with human scores is that the standardized mean score difference (standardized on the distribution of human scores) between the human and computer cannot exceed .15. The standardized difference of the mean is formalized as follows:

$$\overline{Z} = \frac{[\overline{X}_{AS} - \overline{X}_H]}{\sqrt{\frac{SD_{AS}^2 - SD_H^2}{2}}} \quad (1)$$

where $\overline{X}_{AS}$ is the mean of the automated score, $\overline{X}_H$ is the mean of the human score, SD_{AS}^2 is the variance of the automated score, and SD_H^2 is the variance of the human score.

3. EXPERIMENTS

3.1 DATASET

We used the Automated Student Assessment Prize (ASAP) dataset[1], hosted by the Kaggle platform, as our experiment data. ASAP is the most widely used dataset to evaluate the performance of AES systems [8]. All the essays provided are already human graded. ASAP dataset consists of eight prompts, with varying score ranges for each prompt. Table 1 highlights the topics of each prompt in ASAP dataset.

Table 1 Prompts in ASAP Dataset

ASAP Dataset	Topics
Prompt 1	The effects computers have on people
Prompt 2	Censorship in the libraries
Prompt 3	Respond to an extract about how the features of a setting affected a cyclist
Prompt 4	Explain why an extract from *Winter Hibiscus* by Minfong Ho was concluded in the way the author did.
Prompt 5	Describe the mood created by the author in an extract from *Narciso Rodriguez* by Narciso Rodriguez
Prompt 6	The difficulties faced by the builders of the Empire State Building in allowing dirigibles to dock there
Prompt 7	Write a story about patience
Prompt 8	The benefits of laughter

3.2 FEATURES EXTRACTION

Each essay is transformed into a 780 dimension of features vector. We extract the essay features into two categories: 12 interpretable features, and 768 dimension of Sentence-BERT vector representation. Table 2 contains the essay features we used to train the scoring model.

Table 2 Essay Features

Type	Description
Interpretable essay features (12 features)	Answer Length (Character counts) Word count Average word length Count of "good" POS n-grams Number of overlapping tokens with the prompt Number of overlapping tokens (including synonyms) with the prompt Number of punctuations Spelling errors Unique words count Prompt – answer similarity score (SBERT representation) Prompt – answer similarity score (BOW representation) Language Errors
Sentence-BERT features (768 dim)	The encoding of the essay using Sentence-BERT pretrained model

3.2.1 Interpretable Essay Features

Six out of the twelve interpretable essay features are extracted from EASE (Enhanced AI Scoring Engine) library[2], written by one of the winners in ASAP Kaggle competition. This features set have been proven to be robust [9]. EASE generates 414-length features. However, we exclude most of the features generated by EASE library that are mostly Bag-of-Words vectors. The other six features extracted from the text are the number of punctuations, the number of spelling errors, unique words count, similarity scores between

[1] https://www.kaggle.com/c/asap-aes

[2] https://github.com/edx/ease

answer and prompt using S-BERT and BOW (Bag-of-Words) vector representations, and the number of language errors.

The grammar feature is measured by the number of good n-grams in the essay. EASE library extracts the essay text into its POS-tags and compares them with a list of valid POS-tag combinations in English. Good POS n-grams are defined as the ones that separate high- from low-scoring essays, determined using the Fisher test [10]. Moreover, we count the number of language errors in an answer using Language Tool Python library[3]. Mechanics in a language include aspects such as the usage of punctuation and the number of spelling errors found in the answer.

The average word length and long words count are used by Mahana et al. to estimate language fluency and dexterity [11]. Larkey also used the number of long words to indicate the complexity of term usage [12]. Unique words count feature is useful to estimate the richness of vocabulary in the answer.

The relevance factor of an answer combines two features from EASE library, which are related to the degree of tokens overlap between the prompt and the answer, including their synonyms. Two additional features are the cosine similarity measurement between the answer and the prompt, both using the Sentence-BERT and the BOW representation.

3.2.2 Sentence-BERT representation

Sentence-BERT, introduced by Reimers and Gurevych (2019), is a modification of pretrained BERT network using Siamese and triplet network [13]. It converts a text into a 768-dimension feature vectors and produces semantically meaningful sentence embedding. The embedding result can then be compared using cosine-similarity.

3.3 MODEL TRAINING

We train the regression models using Gradient Boosting algorithms, with 80% training data and 20% testing data (using 5-fold cross-validation). We use Quadratic Weighted Kappa (QWK) [14] score as the evaluation metric, which measures the agreement between system predicted scores and human-annotated scores. QWK is the standard evaluation metric to measure the performance of an AES system [1]. Although ASAP dataset has 8 prompts, we trained 9 models in total. The reason is that prompt 2 was scored in two different domains (Writing Application and Language Conventions). Therefore, we must create two separate predictions for this essay prompt. To train all nine models, we used different hyperparameters for each model.

4. RESULTS

4.1 Model Performance Evaluation

In this subsection, we evaluated the model performance based on the quantitative evaluation criteria as discussed in Section 2. Furthermore, we also analyzed the distribution of overall holistic scores assigned by human raters and computer.

4.1.1 Score Agreement Evaluation

We conducted the quantitative evaluation of our models based on the acceptance criteria by Williamson et al. in [1], and the results are shown in Table 3. The table describes the performance measurements for the scoring model of each prompt. The first column is the QWK score, which measures the human-computer agreement. The human-human agreement score in the second column is used to calculate the degradation value. The last column contains the standardized mean score difference between human and automated scores.

Table 3 Model Performance Evaluation for ASAP Dataset

Dataset	QWK Score	Human Agreement	Degradation	$\overline{Z}$
1	0.7826	0.72095	-0.06165	0.0056
2_dom1	0.6731	0.81413	0.14103	0.0007
2_dom2	0.6715	0.80175	0.13025	0.0394
3	0.6887	0.76923	0.08053	0.0272
4	0.7736	0.85113	0.07753	0.0094
5	0.8065	0.7527	-0.0538	0.0229
6	0.7985	0.77649	-0.02201	0.0102
7	0.7771	0.72148	-0.05562	0.0023
8	0.6668	0.62911	-0.03769	0.0147

Based on the above results, we concluded that five models (prompt 1, 4, 5, 6, and 7) satisfy the quantitative evaluation criteria defined as a standard in [1]. Next, we continued our analysis and the reliability and validity tests on only these well-performing models and ignored the other underperforming models (prompt 2_dom1, 2_dom2, 3, and 8).

4.1.2 Distribution of overall holistic scores

We investigated the distribution of the overall holistic scores assigned by human raters and the automated scorer. It is important to understand the distribution of the scores, especially in relation to the decision of an exam. For this purpose, we presented the decision into three categories: passing, borderline, and failing. Table 4 shows the rubric score and resolved score range in ASAP dataset, in which our model passed the quantitative performance evaluation in the previous subsection. The resolved score is the final score after combining the rubric scores from two human raters. Each prompt has a different score resolution. In some prompts, if there was a difference between scorer 1 and scorer 2, the final score was always the higher of the two. In another prompt, the final score was the sum of scores from rater 1 and rater 2 if their scores are adjacent. If non-adjacent, an expert scorer will determine the final score.

Table 4 ASAP Rubric and Resolved Score Range

Dataset	Rubric score	Resolved Score	Borderline Score
Prompt 1	1 - 6	2 - 12	6-8
Prompt 4	0 - 3	0 - 3	1
Prompt 5	0 - 4	0 - 4	2
Prompt 6	0 - 4	0 - 4	2
Prompt 7	0 - 12	0 - 24	12

To categorize the exam results, we created a borderline score from the resolved score. It is not necessarily the exact middle score because some datasets do not have it. We considered scores below the borderline as failing and scores above the borderline as passing.

[3] https://github.com/jxmorris12/language_tool_python

Table 5 Frequency Comparison for Passing, Borderline, and Failing (in %)

	ASAP 1			ASAP 4			ASAP 5			ASAP 6			ASAP 7		
	HR1	**HR2**	**AES**	**HR1**	**HR2**	**AES**	**HR1**	**HR2**	**AES**	**HR1**	**HR2**	**AES**	**HR1**	**HR2**	**AES**
Passing	35.2	35.5	50.1	39.6	40	45.2	37.8	39.2	45.9	59.4	59.1	65	74	72.8	82
Borderline	62.7	62.5	48.8	42.7	41.9	42.3	38.5	36	38.7	25.7	25.9	25.6	8.9	9.8	4.9
Failing	2.1	2	1.1	17.7	18.1	12.5	23.7	24.8	15.4	14.9	15	9.4	17.1	17.4	13.1
Total	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**

As can be seen from Table 5, the use of automated scorer will put the students at an advantage, compared to when the exam only employs human raters. In all datasets, we have the same overview. The AES models assigned scores with much higher passing rates than both human raters. And as far as the failure rates are concerned, the scores from AES indicate very low failing rates, compared to human raters. This finding supports the result in [15], with a different dataset for the experiment.

4.2 Evaluating and Improving the Model

We examined the performance of the model using three scenarios: gibberish answer, paraphrased answer, and off-topic answer. We discuss the evaluation results and strategies to improve the system in the following sections.

4.2.1 Off-topic essay

One way to validate the use of an automated essay scoring system is by checking its performance against off-topic answers. For this study, we use ASAP dataset which has eight sub datasets (prompts). To simulate the experiment, for each model, we used the answers in the other seven prompts as the off-topic essays. We randomly sampled 50 essays from each dataset, resulting in a total of 350 off-topic essays. Using 5-fold cross-validation, we measured the accuracy of the model in predicting the score of the off-topic essays, which we assume should get 0 (zero), due to the complete irrelevance with the corresponding prompt.

Table 6 Accuracy of off-topic detection

Dataset	Training Data	Accuracy (%)	QWK Score
Prompt 1 (2 – 12)	Original	0%	0.7826
	Original + 350 off-topic	55.4%	0.7031
Prompt 4 (0 – 3)	Original	4.3%	0.7736
	Original + 350 off-topic	91.4%	0.7697
Prompt 5 (0 - 4)	Original	0.6%	0.8065
	Original + 350 off-topic	88%	0.7951
Prompt 6 (0 – 4)	Original	5.80%	0.799
	Original + 350 off-topic	97%	0.787
Prompt 7 (0 - 24)	Original	0%	0.7771
	Original + 350 off-topic	45.7%	0.7225

We also investigated the change of value of the QWK scores after retraining the model. The motivation is, we want to avoid that the retraining process degrades the performance of the original model. The new model should still perform well in predicting the original essay set.

Table 6 describes the experiment results of the retraining process, the improvement in accuracies, and the change in QWK scores. The models trained with only the original dataset performed with very low accuracies. The highest result is by prompt 6, with 5.8% of the off-topic essays are correctly given 0 scores. Prompt 1 and prompt 7 are the worst with no correct prediction at all. It means all off-topic essays are graded with scores greater than 0.

We can observe the effect of including the off-topic essays in the training data. The model performance for prompt (4, 5, and 6) drastically increase. For prompt 6, the accuracy on predicting the unseen data of the off-topic essays (test set) reached 97%, with only a slight decrease on the QWK score. There are moderate improvements for prompt 1 and especially prompt 7. We assume this is due to a larger score range, which for prompt 7 is 0 – 24. If we are being less strict about the score 0 (zero) policy for off-topic essays, for example by categorizing score between 0 – 3 as failed score, we obtain a much better accuracy, which is 81.7%. Meanwhile, for prompt 1, the lowest resolved score is 2. However, we trained the model to give off-topic essays 0 score prediction. If we create a score range 0 – 2 as failed category, the accuracy increases to 93%. Because we have many score predictions for the off-topic answers ranging from 0 to 2 by prompt 1.

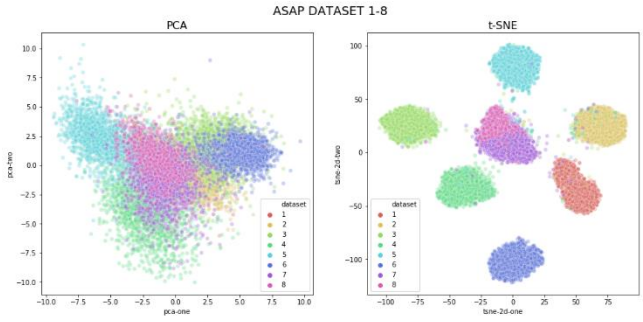


Figure 1 SBERT representation for 8 prompts in ASAP dataset

We conclude that the solution for detecting off-topic essays is relatively simple. Without using additional features for detecting off-topic essays, we found out that the SBERT features are very helpful to be used as features for training a scoring model. Using PCA and t-SNE, we plotted the SBERT vector representation of the essays in all eight of ASAP dataset prompts. Figure 1 shows that using t-SNE, all prompts are almost perfectly separated, although we can see both dataset 7 and dataset 8 are close to each other in the middle of the plot. We assume that it is caused by their similar prompt topics. If we check Table 1 for the description of topics in ASAP dataset, in prompt 7 the students are asked to write a story about patience, while prompt 8 discusses about the benefits of laughter. Both topics are arguably more closely related to each other than when we compare them with the topics of prompt 1 – prompt 6.

4.2.2 *Gibberish*

For the next input scenario, we want to avoid that the system receives invalid answers such as gibberish, and undeservedly returns scores other than zero. Ideally, any gibberish answer must get the score zero. However, using our model, we tested several gibberish as the answers, and the scores are not zero. We provide some examples of the inputs as shown in Table 7. In this table, we show the examples of wrong predictions by the scoring model that was trained using ASAP dataset prompt 6. Nevertheless, we can also observe similar problems in the other models.

Table 7 Examples of Wrong Predictions by Prompt 6

Answer	Score (0-4)
asdafafdf adjhgladghad	1
Eyoqtuwrpituauoyeqo ngbambgagadhkq3124 31794613 hbfka df	1
orkesbroh	1

To analyze the reason, we conducted local model interpretation, which means that we are interested in understanding which variable, or combination of variables, determines the specific prediction. We use SHAP (SHapley Additive exPlanations) values to help in determining the most predictive variables in a single prediction [16]. In AES, the system output is a real number. Each variable contribution will either increase or decrease the output value. One implementation of SHAP libraries is TreeExplainer, a faster library for obtaining SHAP values for tree ensemble methods [17].

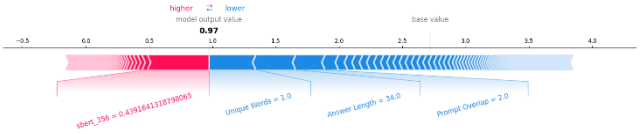


Figure 2 Local interpretation for a single prediction

From Figure 2, it seems that some of the most important interpretable features (number of unique words, answer length, and prompt overlap) have the correct effects to the prediction, they play a role in decreasing the score. However, there is one feature from the SBERT vector representation that helps the score of gibberish answers to increase, i.e. sbert_356. Most of the wrong gibberish predictions have a similar explanation to the one shown in Figure 2. SBERT vector is not interpretable, therefore we cannot explain and analyze the reason of this peculiar model behavior. We propose two solutions to this problem as follows:

1. Retrain model using gibberish data, with label score zero.

We created 200 gibberish essays and transformed them into a 780-dimension vector representation of text, and then include them in the training and testing data. Table 8 shows the performance comparison of three model training scenarios. We can observe a large improvement on the accuracy of the model to detect gibberish answers, and to punish them with the scores zero. For example, by prompt 1, the accuracy increases from 0% to 92.8% by adding only 100 gibberish data to the model training. By adding 100 more data, the accuracy only improves by a little less than 2%. After changing the training data by adding gibberish for the training process, we want to make sure that the main performance metrics (QWK score) on the original data is not being sacrificed. The results show that in all prompts, the addition of gibberish data to the training phase did not harm the performance of the models. The QWK scores decreased by a very small margin, and they still have the human-computer agreement score above the required threshold. Even in prompt 7, the final QWK score increased with the addition of gibberish to the training set.

Table 8 Accuracy of Gibberish Detection

Dataset	Training Data	Accuracy (%)	QWK Score
Prompt 1	Original	0	0.7826
	Original + 100 gibberish	92.8	0.7768
	Original + 200 gibberish	94.4	0.7683
Prompt 4	Original	18.6	0.7736
	Original + 100 gibberish	97.8	0.779
	Original + 200 gibberish	98.6	0.7749
Prompt 5	Original	3.5	0.8065
	Original + 100 gibberish	98	0.7986
	Original + 200 gibberish	98.6	0.8049
Prompt 6	Original	18.2	0.799
	Original + 100 gibberish	96	0.794
	Original + 200 gibberish	97.9	0.782
Prompt 7	Original	0	0.7771
	Original + 100 gibberish	68	0.7798
	Original + 200 gibberish	73.4	0.781

2. Use rule-based mechanism.

This is arguably a simpler solution, without the need to involve any additional model retraining process and could be a more generalizable solution. The system is configured to automatically give score zero for possible gibberish answer, this can be done automatically for example with a valid English word detection library. If none of the token in the answer is valid English vocabulary, we can consider the answer as gibberish. The main drawback is possibly the added processing time for the program to validate each word in the answer, depending on how large the vocabulary is.

4.2.3 *Paraphrased Answer*

To further evaluate the performance of the system, we investigated the reliability of the system by testing whether the model consistently gives the same score for the same answer. For this experiment, we generated paraphrased answers of all answers in the dataset. And we examine whether the model would predict the same score for each paraphrased answer. We utilized an online paraphrasing tool[4] to generate the paraphrased version of the answer.

We use Quadratic Weighted Kappa (QWK) to compute the agreement between the original test set prediction and the

[4] https://spinbot.com/

paraphrased test set prediction. Based on the scores in Table 9, it is evident that the agreements for all datasets are high. Therefore, we conclude that the models perform consistently in predicting the scores of paraphrased answers.

Table 9 Agreement of Prediction between Original and Paraphrased Answers

Dataset	QWK
Prompt 1	0.8109
Prompt 4	0.8674
Prompt 5	0.8645
Prompt 6	0.846
Prompt 7	0.9411

The highest agreement score is achieved by the model of prompt 7, which shows a near perfect agreement with QWK score of 0.9411. Although not as high as the result of prompt 7, the QWK scores in prompt 1, 4, 5, and 6 are considered as very high agreement.

5. CONCLUSION

The purpose of this research is to highlight the limitations of the current performance measurement standard for automated essay scoring. A quantitatively well-performing model with high human – automated score agreement rate, is not necessarily ready for deployment in the real-world usage. We demonstrated that such models still possess some performance concerns against varying input scenarios. We showed empirical evidence that those models have some difficulties, proven by very low accuracies, in detecting off-topic essays and gibberish. We also proposed and proved several strategies that can successfully improve the performance of the system. In another scenario, for consistency testing, the models already performed quite well for predicting paraphrased answers, judged from high agreement results with the predictions on the original answer. While we are aware that there remain more validity questions to be studied, this research can serve as additional techniques towards a better holistic evaluation framework for AES.

6. References

[1] D. M. Williamson, X. Xi and F. J. Breyer, "A Framework for Evaluation and Use of Automated Scoring," *Educational Measurement: Issues and Practice*, vol. 31, pp. 2-13, 2012.

[2] R. E. Bennett and I. I. Bejar, "Validity and automad scoring: It's not only the scoring," *Educational Measurement: Issues and Practice*, vol. 17, p. 9–17, 1998.

[3] Y. Attali and J. Burstein, "Automated essay scoring with e-rater® V. 2," *The Journal of Technology, Learning and Assessment*, vol. 4, 2006.

[4] B. E. Clauser, M. T. Kane and D. B. Swanson, "Validity Issues for Performance-Based Tests Scored With Computer-Automated Scoring Systems," *Applied Measurement in Education*, vol. 15, pp. 413-432, 2002.

[5] M. K. Enright and T. Quinlan, "Complementing human judgment of essays written by English language learners with e-rater® scoring," *Language Testing*, vol. 27, pp. 317-334, 2010.

[6] C.-F. E. Chen and W.-Y. E. C. Cheng, "Beyond the design of automated writing evaluation: Pedagogical practices and perceived learning effectiveness in EFL writing classes," *Language Learning & Technology*, vol. 12, p. 94–112, 2008.

[7] D. E. Powers, J. C. Burstein, M. Chodorow, M. E. Fowles and K. Kukich, "Stumping E-Rater: Challenging the validity of automated essay scoring," *ETS Research Report Series*, vol. 2001, p. i–44, 2001.

[8] J. Liu, Y. Xu and Y. Zhu, "Automated essay scoring based on two-stage learning," *arXiv preprint arXiv:1901.07744*, 2019.

[9] P. Phandi, K. M. A. Chai and H. T. Ng, "Flexible domain adaptation for automated essay scoring using correlated linear regression," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[10] R. A. Fisher, "On the interpretation of χ 2 from contingency tables, and the calculation of P," *Journal of the Royal Statistical Society*, vol. 85, pp. 87-94, 1922.

[11] M. Mahana, M. Johns and A. Apte, "Automated essay grading using machine learning," *Mach. Learn. Session, Stanford University*, 2012.

[12] L. S. Larkey, "Automatic essay grading using text categorization techniques," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998.

[13] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[14] J. Cohen, "Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit.," *Psychological bulletin*, vol. 70, p. 213, 1968.

[15] J. Wang and M. S. Brown, "Automated essay scoring versus human scoring: A comparative study.," *Journal of Technology, Learning, and Assessment*, vol. 6, p. n2, 2007.

[16] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, Eds., Curran Associates, Inc., 2017, p. 4765–4774.

[17] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal and S.-I. Lee, "From local explanations to global understanding with explainable AI for trees," *Nature Machine Intelligence*, vol. 2, pp. 2522-5839, 2020.

Predicting Student Performance Using Teacher Observation Reports

Menna Fateen
Kyushu University
menna.fateen@m.ait.kyushu-u.ac.jp

Tsunenori Mine
Kyushu University
mine@ait.kyushu-u.ac.jp

ABSTRACT

Studying for entrance examinations can be a distressing period for numerous students. Consequently, many students decide to attend cram schools to assist them in preparing for these exams. For such schools and for all educational institutes, it is necessary to obtain the best tools to provide the highest quality of learning and guidance. Performance prediction is one tool that can serve as a resource for insights that are valuable to all educational stakeholders. With accurate predictions of their grades, students can be further guided and fostered in order to achieve their optimal learning goals. In this regard, we target middle school students to be able to guide them on their educational journey as early as possible. We propose a method to predict the students' performance in entrance examinations using the comments that cram school teachers made throughout the lessons. Teachers in cram schools observe their student's behavior closely and give reports on the efforts taken in their subject material. We show that the teachers' comments are qualified to construct a tool that is capable of predicting students' grades efficiently. This is a new method because previous studies focus on predicting grades mainly using student data such as their reflection comments or earlier scores. Experimental results show that using readily available feedback from teachers can remarkably contribute to the accuracy of student performance prediction.

Keywords

text mining, student grade prediction, teacher observation reports, machine learning

1. INTRODUCTION

"If you could reinvent higher education for the twenty-first century, what would it look like?". A question like this one invites many observations about the advantages and issues that the current state of higher education has in the world. As a matter of fact, this question has been addressed specifically by the founders of the Minerva Schools at KGI [1] in the United States. At such innovative universities and schools, active learning and student engagement with the material are highly encouraged [2, 3, 4, 5, 6]. Additionally, the student/teacher ratio is expected to be lower than in traditional schools for higher teacher effectiveness [7]. Students are assessed and observed closely by their teachers and they can receive written feedback from their teachers daily. These reports clarify any confusion, reinforce strong points and give more specific advice and guidance [8, 9]. Besides, since teachers frequently engage with students, research has proven that these teachers, especially those with professional development, can accurately judge and forecast their students' computational skills [10].

In this paper, we propose a novel method for predicting students' performance or final grades. We show that we can use reports carefully written by teachers that closely observe the students, to construct a grade prediction model. If these predictions can be made accurately, it would be an invaluable resource to help the teachers better regulate their students' learning. Future performance prediction is considered a powerful means that can provide all educational stakeholders with insights that are beneficial to them. Many grade prediction models have been proposed by researchers in the last decade [11, 12, 13], but no model has used teacher reports as far as we know. The teacher reports we use are provided by a cram school in Japan. Cram schools are specialized in providing extra and more attentive education for students who want to achieve certain goals, particularly studying for high school or university entrance exams [14]. To capture the meanings of the teacher reports, we obtain vector representations by applying the term-frequency inverse document-frequency (TF-IDF) method and extracting BERT embeddings. Our model uses these vectorized reports as the explanatory variables for a Gradient Boosting regressor. The regressor then predicts the students' scores. Our experiment results show that when adding teachers' reports to the regular student exam scores, we can predict their letter grade with an accuracy up to 62%. To sum up, our contributions can be outlined as follows:

- We propose a new performance prediction method using teacher observation reports represented using TF-IDF and BERT.
- We conducted 2 main different models of prediction and compared the experiment results to show that using teacher reports has the potential to contribute to an increase in accuracy of grade prediction models.

All in all, to the best of our knowledge, this is the first study to use NLP to mine teacher observation comments

Menna Fateen and Tsunenori Mine "Predicting Student Performance Using Teacher Observation Reports". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 481-486. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

to predict student grades. Our research and experimental results demonstrate the potential that these unstructured teacher observation comments have in predicting students' total scores and final letter grades.

2. RELATED WORK

The utilization of data mining and machine learning or deep learning tools to construct predictive models are increasingly being adopted in many different fields [15, 16]. Needless to say, the educational field has not been an exception. Topics in educational data mining vary widely from course recommendation systems [17] to automatic assessment [18]. More specifically, an extensive amount of studies have been dedicated to prediction modeling whether it be predicting student grades or performance such as next-term grade prediction [19] or student dropout. These prediction models are essential since they underlie applications to important educational AI-based decision-making systems [20]. With accurate predictions, the performance of students can be monitored using these systems, and students that have difficulties in their studies can easily be detected and given further guidance early on.

Over the past years, several methods have been developed to predict student's performance using Natural Language Processing (NLP) techniques. It has been proven that mining unstructured text using NLP has the capacity to contribute to accurately predicting students' success over the information obtained from usual fixed-response items [21]. Luo et. al [13] proposed a method to predict student grades based on their free-style reflection comments collected after each lesson. The comments were collected according to the PCN method [22] that categorizes the students' comments. To represent the students' reflection comments, Word2Vec embeddings were adopted followed by an artificial neural network. Their experiments show a correct rate of 80%. Teacher or advisor notes have been used by Jayaraman, not to predict student grades, but to detect students that are at risk of dropping out of college [23]. In their study, they use sentiment analysis to extract the positive and negative sentiment from the advisors' notes and use those as features to train a model. The model achieves 73% accuracy at identifying at-risk students.

3. DATA DESCRIPTION

The dataset obtained and used for our model was provided by a cram school in Fukuoka, Japan. To ensure confidentiality, no student names or other identifying data were presented. Reports were obtained monthly and sent as CSV files. Since our model is focused on predicting the performance of students in their entrance examinations, we focused on those students in their final year of middle school. The final dataset after preprocessing composed of 11,960 reports over the period from May to October for 159 students.

3.1 Monthly Reports

In addition to the student ID and the class date, each report also consisted of the subject code, the teacher's comments, understanding, attitude and homework scores. More data in the reports were also provided but were unstructured and considered redundant for the prediction model. The features that were extracted from the reports and used in the study are discussed in more detail in Section 4.1. However our main explanatory variable used in the study is the teachers' observation comments written in Japanese. The average length of these comments is 96 characters. In addition, by analyzing comments, it was observed that teachers tend to encourage and energize their students by using words such as "better" and "work on". Moreover, the words used in the comments depend on the context or class subject to some degree. For example, the expression "calculation problem" is likely to be used in math lessons.

Table 1: Number of Reports in Each Subject

	Japanese	Math	Science	Social	English
Number of Reports	1157 (9.7%)	3547 (29.6%)	2428 (20.3%)	1669 (14%)	3159 (26.4%)

In the cram school, students take different lessons for each subject. These lessons fall under the 5 main subjects: Japanese, Mathematics, Science, Social Studies and English. Since the main objective of our model is to predict a student's total score, reports in all 5 subjects are required. Therefore, testing the model was only possible for those students who attended classes for all subjects. The number of reports that fall under each subject are shown in Table 1. The values in the table show that the most taken lessons and therefore the most reports provided were in the subject of Mathematics followed directly by English. The number of total reports for each student varied depending on the classes attended. The average number of total reports recorded for each student was 82 reports with a maximum of 206 and a minimum of 24 reports.

3.2 Test Scores

Students attending the cram school were naturally registered in many different schools. The results of their regularly taken examinations at school were recorded and provided. These scores were what we considered student data and would be traditionally used as the main feature to predict their performance in the entrance exam. To teach the model to perform these predictions, we adopted the supervised learning method. In supervised learning, training data needs to be labeled with the required outputs for each input. This enables the model to train its learning function by altering it based on the correct result so that the function can then be applied to new inputs. In our study, we used the students' results in their cram school simulation exams as the labels for the model since their actual performance in the entrance exam was unattainable.

The simulation scores for the 159 students were recorded for all subjects and also provided as the total score. To visualize the distribution of the students' scores, histograms were plotted as shown in Figure 1. The shape of the graph for the subject scores distribution and total score distribution is approximately bell-shaped and seems symmetric about the mean, so it is assumed that the scores follow the normal distribution. The standard deviation, σ, for all scores are displayed in Table 2 to show how dispersed the values are.

4. METHODOLOGY

4.1 Feature Selection

Figure 1: Distribution of Simulation Test Scores

Table 2: Standard deviation of subject scores

	Japanese	Math	Science	Social	English	Total
σ	11.85	16.62	20.33	16.93	18.47	70.01

For our experimental settings, we adopt 3 main feature sets for the sake of comparison. The first feature set, FS_1, consists of using teachers' report contents as the main explanatory variables. A teacher's report in one lesson evaluating the student consists of 1-Comments 2- Understanding Score 3- Attitude Score and 4-Homework Score. We use all of these attributes except for the homework score. This is mainly because more than 36% of the reports did not include homework scores since not all lessons necessarily require homework. After each lesson, the teacher writes some comments based on their observations, assesses the student on their understanding giving them a score of either (0-30-60-80-100) and an attitude score of either (1-2-3-4). The second feature set, FS_2, consists of student-related data only, specifically their gender and the score of their regularly scheduled exam at school. Since we predict each subject score separately, the regular score corresponds to the subject score. As for the students' gender, the Pearson correlation coefficient between it and the score is 0.12 while the correlation coefficient between the regular score and the simulation score is 0.80 which suggests that the important factor in FS_2 is essentially the student regular score and not the gender. Finally, we investigate using both teachers' reports and the regular student scores to verify whether adding teachers' reports contributes to the accuracy of the prediction model or not. The third feature set, FS_3, is essentially a concatenation of FS_1 and FS_2. A sample of FS_1 is shown in Table 3.

4.2 Natural Language Processing

There are numerous ways to represent text data for a machine learning model to convey the original meanings of the text and prevent information loss. In our experiments, we chose to represent the teachers' comments using two techniques. We used the traditional TFIDF vectorization method and compared it with BERT embeddings.

4.2.1 TF-IDF

The first essential step in transforming text into a numerical representation is preprocessing the text. This step begins with tokenization or splitting the sentences into words. Tokenization in languages such as English can be done by splitting the sentence strings at each space. However, for Japanese, this step is merged with the next, which is morphological analysis, since there are no spaces in Japanese sentences. We use the fugashi [24] parser for this step, which is essentially a wrapper for Mecab[1], a Japanese tokenizer and morphological analysis tool. Our parser extracts from each report the following parts of speech: nouns, verbs, auxiliary verbs, adjectives and adverbs. We use the corresponding terms to these extracted parts of speech to build a bag-of-words vector with weights given by the TF-IDF method implemented by sklearn [25]. Since the teachers' comments are given in Japanese, we provide the mentioned parser to the tokenizer parameter. We also give a list of predefined Japanese stop words to the vectorizer.

4.2.2 BERT

BERT or Bidirectional Encoder Representations from Transformers is a new method of pre-training language representations presented by Google [26]. BERT obtains state-of-the-art results on many NLP tasks. It is a Transformer Encoder stack that pre-trains language representations. A pre-trained BERT model is basically a general purpose language understanding model trained on a large corpus which can then be used for downstream tasks. The BERT model we used for the comments was pretrained by Inui Laboratory, Tohoku University[2]. The corpus they used for pretraining was Japanese Wikipedia and the model was trained with the same configuration as the original BERT. In the experiments shown in this paper, we used the BERT [CLS] token embeddings as our BERT embeddings.

4.3 Evaluation Metrics

To evaluate our experiments, we use the Mean Absolute Error (MAE) metric. The MAE is calculated using the following formula :

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\text{score}_{pred,i} - \text{score}_{true,i}| \qquad (1)$$

where $\text{score}_{true,i}$ is the actual score that student i obtained. The predicted score ($\text{score}_{pred,i}$) is calculated differently for subject scores and total score. For a specific subject $s \in$ S, where S = {Japanese, Math, Science, Social Studies, English}, a student i can attend a variable number t of lessons. Therefore, to predict the subject score ($\text{SubjectScore}_{pred,i,s}$) of student i we use each of their reports as independent inputs to the model and obtain an ordered list $X_{i,s,t}$ of pre-

[1] `https://taku910.github.io/mecab/#parse`
[2] `https://github.com/cl-tohoku/bert-japanese`

Table 3: A sample of FS_1: teachers' reports (comments originally in Japanese)

Understanding	Attitude	Comments
80	4	We are trying applied problems of resolution into factors. You look like making many mistakes carelessly, but know formulas very well.
80	4	We are trying applied problems of resolution into factors. You look like making many mistakes carelessly, but know formulas very well.
100	4	He took notes while watching the commentary and focused on the problem. If you keep going at this rate, you will be able to meet the target, the 5th time. So, let's do our best!

dicted scores for student$_i$. The estimated score for the subject is then decided using:

$$\begin{aligned}\text{SubjectScore}_{pred,i,s} &= Med(X_{i,s,t}) \\ &= \begin{cases} X_i[\frac{t}{2}], & \text{if t is even} \\ \frac{1}{2}(X_i[\frac{t-1}{2}] + X_i[\frac{t+1}{2}]), & \text{if t is odd} \end{cases}\end{aligned} \tag{2}$$

To measure the central tendency, we used the median rather than the mean as it is robust to skewness and outliers. Nevertheless, if the estimations follow a normal distribution, the median would be close to the mean. The total predicted score ($\text{TotalScore}_{pred,i}$) can then be estimated by:

$$\text{TotalScore}_{pred,i} = \sum_{s \in S} \text{SubjectScore}_{pred,i,s} \tag{3}$$

Finally, since students receive letter grades for their total score, we map the estimated total score to its closest corresponding letter grade according to the percentages shown in Table 4 [27]. We then compute the percentage of grades that are x ticks away from their actual grades. A tick, as specified by [28], is defined as the difference between two successive letter grades. We name this metric percentage by tick accuracy or PTA. PTA_0 stands for the Percentage by 0 Tick Accuracy which means the model successfully predicted the letter grade with no error while PTA_1 is the percentage of incorrectly predicted grades but are 1 tick away from the true letter grade (e.g. A vs B). A similar metric was used in previous studies regarding grade prediction models [11, 28].

Table 4: Letter grades and their corresponding percentages

Grade	S	A	B	C	D	F
%	90-100	80-89	70-79	60-69	50-59	0-49

5. EXPERIMENTS

5.1 Model Overview

In our experiments, we adopt gradient boosting, a composite machine learning algorithm. We employed its sklearn implementation, `GradientBoostingRegressor` [25] to predict the continuous value of the students' scores in each subject. Since there is no prior research on the effect of using teacher observation reports in predicting students' grades, we use the following method as the baseline in our experiment. At first, subject codes were unavailable for each teacher observation record. Therefore, we constructed a model that used all of each student's reports, regardless of the subject, to directly predict and estimate the total score according to Equation 2. We call this model, the 'Direct' model. Subject codes then became accessible and we were able to map each report to its corresponding subject. Leveraging that, we created a separate regression model for each subject's reports and estimated the total score as shown in Equation 3. This model is called the 'Subjects' model.

5.2 Experimental Results

All experiments in the study were evaluated using group 10-fold cross-validation. The advantage of group k-fold cross validation method is that all data are used for both training and testing, and each instance is used for testing once. This is especially useful in situations where data is limited. Since the dataset comprises reports for 159 students, we used 143 students' reports for each fold as the training set and 16 as the testing set. The number of reports or instances for each subject model, therefore, varied depending on how many lessons each student had attended. The average MAE, which is calculated as in Equation 1, of all ten folds was computed and used as the main evaluation metric. We ran the baseline Direct model with the 3 feature sets described in Section 4.1. Teachers' comments were represented using BERT embeddings. The performance results are shown Using all 3 feature sets, the Subjects model consistently outperforms the Direct baseline model. Specifically, predicting the total score using the Subjects model with FS_3, which uses both teachers' reports and student data, resulted in a decrease in MAE of 5.62. Using teachers' reports alone (FS_1) resulted in a comparatively higher MAE in both models. However, adding teachers' reports to student data (FS_3) showed a smaller value in MAE than using student data only (FS_2) which suggests that teachers' reports as features can contribute to the accuracy of the grade prediction model.

Table 6 shows the MAE, PTA_0 and PTA_1 of each subject's score prediction model. We ran the subject model with all 3 feature sets. For FS_1 and FS_3, we compared the performance of the two text representations, TF-IDF vectors and BERT embeddings. Values in bold indicate the leading scores for each metric in all subjects. In terms of MAE, using FS_3 consistently outperforms the other feature sets. It can also be seen that BERT embeddings tend to have better overall

Table 5: Average MAE of total score prediction with Direct model vs Subject model using the 3 feature sets: FS_2: student data, FS_1: teacher reports, FS_3: FS_1 + FS_2

	FS_2	FS_1	FS_3
Direct	42.73	53.81	38.91
Subjects	**36.83**	**52.02**	**33.29**

Table 6: Evaluation metric scores in all subjects using the 3 feature sets and comparing between using TFIDF for text representation vs using BERT embeddings. Values in bold indicate the best metric value in a specific subject.

		Japanese			Math			Science			Social Studies			English			Total		
		MAE	PTA_0	PTA_1	MAE	PTA_0	PTA_1	MAE	PTA_0	PTA_1	MAE	PTA_0	PTA_1	MAE	PTA_0	PTA_1	MAE	PTA_0	PTA_1
	FS_2	10.32	0.37	0.20	10.96	**0.53**	0.12	15.02	**0.49**	0.12	13.43	0.51	0.087	12.48	0.58	0.12	36.83	0.58	0.15
TFIDF	FS_1	9.79	0.36	0.22	12.53	0.47	0.10	17.25	0.37	0.09	13.57	**0.60**	0.01	14.93	0.52	0.00	54.81	0.47	0.07
	FS_3	**9.16**	**0.38**	0.20	10.37	0.50	0.16	14.07	0.44	0.16	12.08	0.56	0.086	12.10	0.58	**0.13**	35.19	0.621	0.14
BERT	FS_1	9.47	0.27	**0.23**	12.36	0.45	0.07	16.66	0.40	0.11	13.92	0.55	0.02	14.51	0.52	0.02	52.02	0.49	0.07
	FS_3	9.32	0.37	0.22	**10.12**	0.52	**0.18**	**13.31**	0.43	**0.18**	**12.00**	0.53	**0.095**	**10.99**	**0.62**	0.11	**33.29**	**0.622**	**0.17**

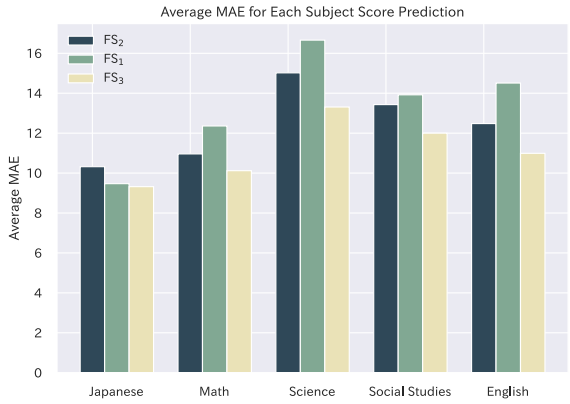


Figure 2: Average MAE of subject scores across all FS

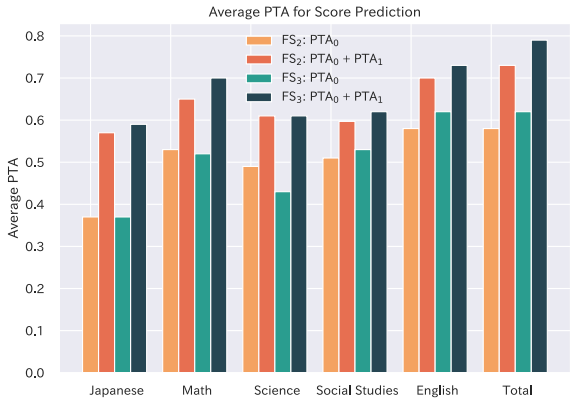


Figure 3: A comparison of PTA metric evaluated when using FS_2 and FS_3 across all subject scores and total score

performance than the TF-IDF vectors. Moreover, running the Subjects model with FS_1 using BERT resulted in lower MAE than when using TF-IDF. Finally, when predicting the total score, using FS_3 with BERT held the top scores across all evaluation metrics.

Figure 2 depicts the performance of each subject seperately in terms of MAE across the three feature sets. It can be observed that FS_3 continuously achieves lower MAE than FS_2 and FS_1. In addition, as shown in Figure 3, FS_3 also consistently achieves higher overall PTA. When predicting the total score, FS_3 shows an increase of 6.2% in $PTA_0 + PTA_1$. These results provide evidence and suggest that teachers' reports can in fact add value and contribute to grade prediction models.

6. DISCUSSION

The results presented in the previous section can be summarized into the following main points.

- The highest performance of the grade prediction model can be achieved by using a concatentation of the two feature sets, FS_1 and FS_2.
- When predicting the total score with teachers' reports, using BERT embeddings outperforms TF-IDF.

The success of BERT can be attributed to the fact that the BERT model has been pretrained on huge corpora of Japanese text data. TFIDF vectors, on the other hand, only use the data on hand to produce the representations. However, an important advantage of TFIDF is that the numerical vector representations are computed much faster than extracting BERT embeddings. To further increase the accuracy of the prediction model considering FS_3 and FS_1, we aim to pre-train BERT on each of the 5 subject reports. It has been proven that pretraining BERT on specific domains can lead to a significant increase in performance [29].

7. CONCLUSION

At educational institutes where students are closely observed by their teachers, large amounts of unstructured data exist in the form of reports and comments. In this paper, we attempted to employ and take advantage of these comments to help identify students that may need extra guidance or attention. Our model used teacher observation comments to predict students' total scores. We applied both TF-IDF and BERT embeddings to the observation comments and used the vectors as inputs to a gradient boosting regressor. Three main feature sets were employed in our model, teacher-related features, student-related features, and a concatenation of both. The performance of our model on each set was then demonstrated. Our experimental results showed that the readily available teachers' reports have the potential to create a grade prediction model. Using teachers' reports can increase the accuracy of a grade prediction model that uses only students' previous exam scores by 6.2%. However, there remains room for improvement in our experiments. We believe that with more teachers' comments, the accuracy of our model could increase. We also plan to enhance the text representations by pretraining BERT on the teachers' comments in advance. Additionally, we intend to experiment with another model architecture that would focus on classifying the students' performance first. We hope that with

such well-defined grade prediction models, we can help guide young students and provide a more focused and personalized education to them.

8. ACKNOWLEDGMENTS

This work was supported in part by e-sia corporation and JSPS KAKENHI Grant Numbers: JP21H00907, JP20H01728, JP20H04300, JP19KK0257, and JP18K18656.

9. REFERENCES

[1] R. Kerrey, *Building the intentional university: Minerva and the future of higher education*. MIT Press, 2018.

[2] T. J. Perry and C. Robichaud, "Teaching ethics using simulations: Active learning exercises in political theory," *Journal of Political Science Education*, vol. 16, no. 2, pp. 225–242, 2020.

[3] M. Hernández-de Menéndez, A. V. Guevara, J. C. T. Martínez, D. H. Alcántara, and R. Morales-Menendez, "Active learning in engineering education. a review of fundamentals, best practices and experiences," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 13, no. 3, pp. 909–922, 2019.

[4] A. Phillipson, A. Riel, and A. B. Leger, "Between knowing and learning: New instructors' experiences in active learning classrooms.," *Canadian Journal for the Scholarship of Teaching and Learning*, vol. 9, no. 1, p. n1, 2018.

[5] J. C. Shin, "University teaching: Redesigning the university as an institution of teaching," 2014.

[6] J. Pirker, M. Riffnaller-Schiefer, and C. Gütl, "Motivational active learning: engaging university students in computer science education," in *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pp. 297–302, 2014.

[7] N. Koc and B. Celik, "The impact of number of students per teacher on student achievement," *Procedia-Social and Behavioral Sciences*, vol. 177, pp. 65–70, 2015.

[8] G. Eyers and M. Hill, "Improving student learning? research evidence about teacher feedback for improvement in new zealand schools," *Waikato Journal of Education*, vol. 10, 2004.

[9] Y. Han and Y. Xu, "The development of student feedback literacy: the influences of teacher feedback on peer feedback," *Assessment & Evaluation in Higher Education*, vol. 45, no. 5, pp. 680–696, 2020.

[10] K. W. Thiede, J. L. Brendefur, R. D. Osguthorpe, M. B. Carney, A. Bremner, S. Strother, S. Oswalt, J. L. Snow, J. Sutton, and D. Jesse, "Can teachers accurately predict student performance?," *Teaching and Teacher Education*, vol. 49, pp. 36–44, 2015.

[11] Z. Ren, X. Ning, A. S. Lan, and H. Rangwala, "Grade prediction based on cumulative knowledge and co-taken courses.," *International Educational Data Mining Society*, 2019.

[12] Y. Zhao, Q. Xu, M. Chen, and G. M. Weiss, "Predicting student performance in a master of data science program using admissions data.," *International Educational Data Mining Society*, 2020.

[13] J. Luo, S. E. Sorour, K. Goda, and T. Mine, "Predicting student grade based on free-style comments using word2vec and ann by considering prediction results obtained in consecutive lessons.," *International Educational Data Mining Society*, 2015.

[14] R. J. Lowe, "Cram schools in Japan: The need for research," *The Language Teacher*, vol. 39, no. 1, pp. 26–31, 2015.

[15] M. Chen, Y. Hao, K. Hwang, L. Wang, and L. Wang, "Disease prediction by machine learning over big data from healthcare communities," *Ieee Access*, vol. 5, pp. 8869–8879, 2017.

[16] M. A. Rushdi, A. A. Rushdi, T. N. Dief, A. M. Halawa, S. Yoshida, and R. Schmehl, "Power prediction of airborne wind energy systems using multivariate machine learning," *Energies*, vol. 13, no. 9, p. 2367, 2020.

[17] A. Esteban, A. Zafra, and C. Romero, "A hybrid multi-criteria approach using a genetic algorithm for recommending courses to university students.," *International Educational Data Mining Society*, 2018.

[18] Z. Wang, A. S. Lan, A. E. Waters, P. Grimaldi, and R. G. Baraniuk, "A meta-learning augmented bidirectional transformer model for automatic short answer grading.," in *EDM*, 2019.

[19] S. Morsy and G. Karypis, "Cumulative knowledge-based regression models for next-term grade prediction," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 552–560, SIAM, 2017.

[20] A. Elbadrawy, A. Polyzou, Z. Ren, M. Sweeney, G. Karypis, and H. Rangwala, "Predicting student performance using personalized analytics," *Computer*, vol. 49, no. 4, pp. 61–69, 2016.

[21] C. Robinson, M. Yeomans, J. Reich, C. Hulleman, and H. Gehlbach, "Forecasting student achievement in moocs with natural language processing," in *Proceedings of the sixth international conference on learning analytics & knowledge*, pp. 383–387, 2016.

[22] K. Goda and T. Mine, "Analysis of students' learning activities through quantifying time-series comments," in *International conference on knowledge-based and intelligent information and engineering systems*, pp. 154–164, Springer, 2011.

[23] J. Jayaraman, "Predicting student dropout by mining advisor notes," in *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, pp. 629–632, 2020.

[24] P. McCann, "fugashi, a tool for tokenizing Japanese in python," in *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, (Online), pp. 44–51, Association for Computational Linguistics, Nov. 2020.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[27] C. Vroman, *Japan: a Study of the Educational System of Japan and Guide to the Academic Placement of Students from Japan in United States Educational Institutions: Placement Recommendations by the Council on Evaluation of Foreign Student Credentials, Meeting July 29-30, 1965*. American Association of Collegiate Registrars and Admissions Officers, 1966.

[28] A. Polyzou and G. Karypis, "Grade prediction with models specific to students and courses," *International Journal of Data Science and Analytics*, vol. 2, no. 3, pp. 159–171, 2016.

[29] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," *arXiv preprint arXiv:2004.10964*, 2020.

Recommending Knowledge Concepts on MOOC Platforms with Meta-path-based Representation Learning

Guangyuan Piao
Department of Computer Science, Hamilton Institute
Maynooth University
Maynooth, Co Kildare, Ireland
guangyuan.piao@mu.ie

ABSTRACT

Massive Open Online Courses (MOOCs) which enable large-scale open online learning for massive users have been playing an important role in modern education for both students as well as professionals. To keep users' interest in MOOCs, recommender systems have been studied and deployed to recommend courses or videos that a user might be interested in. However, recommending courses and videos which usually cover a wide range of knowledge concepts does not consider user interests or learning needs regarding some specific concepts. This paper focuses on the task of recommending knowledge concepts of interest to users, which is challenging due to the sparsity of user-concept interactions given a large number of concepts. In this paper, we propose an approach by modeling information on MOOC platforms (e.g., teacher, video, course, and school) as a Heterogeneous Information Network (HIN) to learn user and concept representations using Graph Convolutional Networks based on user-user and concept-concept relationships via meta-paths in the HIN. We incorporate those learned user and concept representations into an extended matrix factorization framework to predict the preference of concepts for each user. Our experiments on a real-world MOOC dataset show that the proposed approach outperforms several baselines and state-of-the-art methods for predicting and recommending concepts of interest to users.

Keywords

User Modeling, MOOC, Learning Analytics, Knowledge Concept, Recommender Systems

1. INTRODUCTION

MOOCs (Massive Open Online Courses), which are free online courses available to anyone to enroll around the world, have gained a lot of popularity in the past decade. By the end of 2018, popular MOOC platforms such as edX[1], and Coursera[2] have provided 11,400 courses with 101 million users/learners on those platforms[3]. Previous studies have shown that MOOCs do have a real impact [24, 8]. For example, Chen et al. [8] showed that 72% of survey respondents reported career benefits and 61% reported educational benefits. Despite of the popularity, one main challenge of MOOCs is the overall completion rate of those courses is normally lower than 10% [19, 30]. Therefore, understanding and predicting user behaviors and learning needs are important to keep users learning on MOOC platforms.

To this end, previous studies have focused on understanding dropout or procrastination behavior [28, 12, 38, 14] and recommending content such as courses and learning paths that a user might be interested in [16, 26, 4]. A MOOC can be seen as a sequence of videos where each video is associated with some knowledge concepts. For example, a video in a computer science MOOC can cover several concepts such as "software" and "hardware". More recently, Gong et al. [13] argued that course or video recommendations overlook user interests regarding specific knowledge concepts. For example, data mining courses taught by different teachers can be quite different in a microscopic view, and a user who is interested in some specific concepts such as "association rules" might be interested in various video clips or learning materials from different teachers covering those concepts from different perspectives. Therefore, understanding a user's learning needs from a microscopic view and predicting knowledge concepts that the user might be interested in are important.

In this work, we focus on predicting and recommending knowledge concepts that might be interesting to users on MOOC platforms. Based on the interaction history between users and concepts (i.e., a user has interacted with a concept if the user has learned that concept), traditional recommendation approaches such as collaborative filtering (CF) — which recommends similar items (concepts) based on a user's interaction history or interesting items from similar users — can be applied. However, the sparsity of user-item (user–concept) relationships can limit the performance of CF-based methods. In addition to users and concepts, MOOC platform data normally contain other entities such as courses, videos, and teachers as well as the relationships among those entities.

To cope with the sparsity problem, we model those enti-

[1] https://www.edx.org/

[2] https://www.coursera.org/

[3] https://bit.ly/3tScITp

Guangyuan Piao "Recommending Knowledge Concepts on MOOC Platforms with Meta-path-based Representation Learning". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 487-494. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

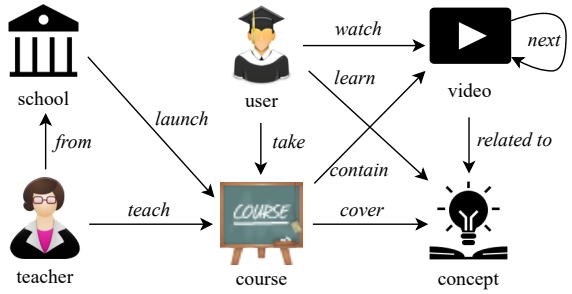


Figure 1: Different types of entities and relationships between two entities in MOOCs. A user u is interested in a concept if u has learned or is going to learn it in the future.

ties and their relationships as a heterogeneous information network (HIN) [33] consisting of the entities and relationships inspired by [13], which can be used for learning user (concept) representations/embeddings by exploring indirect user-user (concept-concept) relationships with Graph Convolutional Networks (GCNs). Figure 1 illustrates such a HIN which we discuss in detail in Section 3. For example, one can derive a homogeneous user graph based on an indirect path in the HIN, e.g., a graph with users and edges between two users if they have taken the same course. Given such a homogeneous graph, traditional GCNs can be applied to the graph to learn the representations/embeddings of users and concepts with respect to the chosen path.

Based on different indirect paths chosen, we can derive various user (concept) representations, and those representations of users (concepts) regarding different paths can be aggregated, e.g., using the mean of those representations. Instead of the straightforward mean aggregation, we propose and investigate different attention mechanisms to derive aggregated user (concept) representations based on different paths. The intuition behind using an attention mechanism is that different paths might have different importance for each user. Afterwards, those learned user and concept representations can be used for predicting the preference scores of concepts for recommendations. Our contributions in this work are as follows: (1) We propose an end-to-end framework[4] for predicting and recommending knowledge concepts of a user's interest in Section 4; (2) We investigate two attention mechanisms for aggregating information from different meta-paths (the definition can be found in Section 3) to derive user and concept representations. We then incorporate those representations into our extended matrix factorization framework for predicting the preference score of a concept with respect to a user; (3) Finally, we evaluate our approach with several baselines and state-of-the-art approaches in terms of well-established evaluation metrics, and show the effectiveness of our proposed approach in Section 6.

2. RELATED WORK

Recommender Systems and User Modeling on MOOC Platforms. There has been growing interest in recommender systems on MOOC platforms since 2013 with respect to different aspects such as course, video, and learning paths [16, 26, 3, 43, 9, 23]. For instance, the authors in [3] proposed YouEDU, which is a pipeline for classifying MOOC forum posts and recommending instructional video clips that might be helpful for resolving confusion detected in those posts. In [21], the authors showed that peer recommendations can improve users' engagement significantly in the context of a Project Management MOOC. Dai et al. [9] proposed analyzing course content for recommending personalized learning paths on MOOC platforms. Khalid et al. [18] provides a comprehensive survey on recent advances regarding different recommender systems in the context of MOOCs. More recently, researchers have started modeling user interests in the context of MOOCs while user modeling has been widely studied in other domains such as social media [42]. For example, Li et al. [22] investigated the impact of acquiring user interests via surveys or questionnaires on course recommendations. In [2], the authors proposed LeCoRe which exploits user interest modeling for recommending courses as well as similar users for promoting peer learning in enterprise environment. Gong et al. [13] argued that course recommendations overlook user interests regarding specific knowledge concepts, and studying users' online learning interests from a microscopic view and recommending knowledge concepts can capture user interests better and provide the flexibility of choosing learning resources of their interest. In this work, we also focus on the microscopic view for knowledge concept recommendations.

Recommendation Approaches with HIN. The basic idea of early recommendation approaches with HIN is to leverage path-based semantic relatedness between users and items over HINs, e.g., leveraging meta-path-based similarities for recommendation [40, 32, 41]. For example, Shi et al. [32] proposed predicting item ratings based on those from similar users measured via different meta-paths. With the advances of graph representation learning, the authors in [31] proposed using pre-trained user and item embeddings based on meta-path information with random walk, and incorporated those pre-trained embeddings as features into an extended matrix factorization framework. The most similar work to ours is Gong et al. [13], which is one of the first works for recommending knowledge concepts on MOOC platforms in a heterogeneous view. The authors showed that their proposed approach outperforms other CF-based baselines as well as metapath2vec [11], which uses learned node representations of a given HIN for knowledge concept recommendations by measuring the similarities between two nodes. Our work differs from [13] in several aspects. First, we formulate interacted concepts for each user as implicit feedback while [13] treated the number of clicks as ratings and formulated the problem as rating prediction for recommending top–k unknown concepts with higher ratings. Secondly, we investigate different attention mechanisms including the one incorporating the latent features of users (items) from matrix factorization. Thirdly, the prediction layer (Eq. 6) for estimating the preference score of a concept is different from [13] which uses the user (item) representations as features for the final prediction.

3. PRELIMINARIES

In this work, we consider the task of predicting and recommending concepts that a user might be interested in based

[4]GitHub repo:`https://github.com/parklize/kgc-rec`

on their learning history, which includes a set of learned concepts and their contextual information such as courses, videos, etc. With n users $\mathcal{U} = \{u_1, \cdots, u_n\}$, and m concepts $\mathcal{C} = \{c_1, \cdots, c_m\}$, we define an implicit feedback matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$ with each entry $r_{u,c} = 1$ if u has learned c and $r_{u,c} = 0$ otherwise. The task can be framed in the context of HIN which is denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consisting of a object set $\mathcal{V}$ and a link set $\mathcal{E}$. A HIN is also associated with an object type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{O}$ and a link type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$. $\mathcal{O}$ and $\mathcal{R}$ denote the sets of predefined object and link types, where $|\mathcal{O}|+|\mathcal{R}| > 2$ [33]. The MOOC data in our study can be represented as a HIN. The HIN consists of six types of entities such as *user*, *concept*, *video*, *course*, *school*, and *teacher*. In addition, there is a set of links describing the relationships among those entities. On top of the definition of HIN, the concept of network schema is used to describe the meta structure of a network [31].

The **network schema** [35] is denoted as $\mathcal{S} = (\mathcal{O}, \mathcal{R})$. It is a meta template for an information network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with the object type mapping $\phi : \mathcal{V} \rightarrow \mathcal{O}$ and the link type mapping $\psi : \mathcal{E} \rightarrow \mathcal{R}$, which is a directed graph defined over object types $\mathcal{O}$, with edges as links from $\mathcal{R}$. Fig. 1 shows the network schema of our MOOC dataset with the six different entity types and the semantic links between them. Given the network schema, we can extract semantic meta-paths between a pair of entities. A meta-path can be formally defined as follows:

A **meta-path** [34] MP is defined on a network schema $\mathcal{S} = (\mathcal{O}, \mathcal{R})$ and is denoted as a path in the form of $O_1 \xrightarrow{R_1} O_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} O_{l+1}$, which describes a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between object O_1 and O_{l+1}, where $\circ$ denotes the composition operator on relations.

4. PROPOSED APPROACH

In this section, we introduce our proposed approach `MOOCIR` (MOOC Interest Recommender) based on meta-paths in the MOOC HIN. In high level, our approach extends the matrix factorization (MF) $\hat{y}_{u,c} = \mathbf{x}_u^T \mathbf{z}_c$, where $\hat{y}_{u,c}$ denotes the predicted preference score of concept c with respect to user u, and $\mathbf{x}_u$ and $\mathbf{z}_c$ refer to *latent features* of u and c, respectively. We extend the MF with user (concept) representations/embeddings that are learned by applying GCNs to meta-path-based graphs. Fig. 2 shows an overview of our approach, which consists of four main components. In the following, we describe each component in detail.

Table 1: Meta-paths selected for extracting user-user and concept-concept relationships.

Type	Meta-path
User	$user \rightarrow concept \xrightarrow{-1} user$
	$user \rightarrow course \xrightarrow{-1} user$
	$user \rightarrow video \xrightarrow{-1} user$
	$user \rightarrow course \rightarrow teacher \xrightarrow{-1} course \xrightarrow{-1} user$
Concept	$concept \rightarrow user \xrightarrow{-1} concept$
	$concept \rightarrow course \xrightarrow{-1} concept$

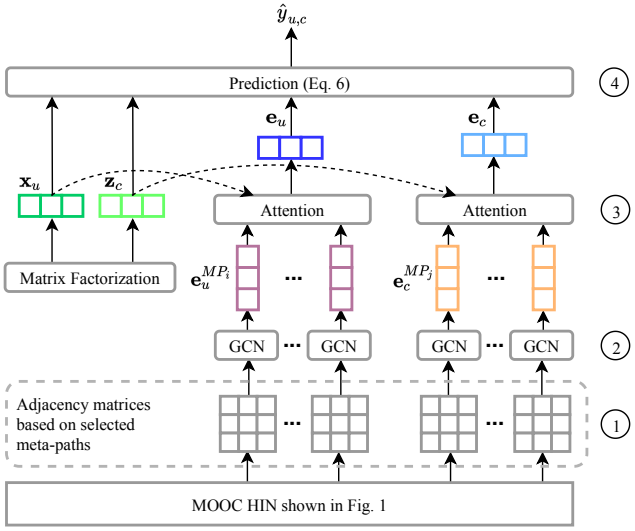


Figure 2: Overview of our proposed approach `MOOCIR`.

Meta-path selection. As discussed in Section 3, meta-paths provide the capability to derive entity-entity relationships through those paths. Similar to previous studies [13, 31], we consider user-user and concept-concept relationships via different meta-paths. To fairly compare with [31] in our experiments, we use the same set of meta-paths used in [31] for our study. Table 1 summarizes six meta-paths used for our work where four paths for users and two for concepts. For each meta-path, a homogeneous graph with respect to users (concepts) can be extracted, which is depicted as its corresponding adjacency matrix in Fig. 2. As one might expect, each entry in the adjacency matrix $\mathbf{A}$ regarding a meta-path is equal to one if two users (concepts) can be connected via that meta-path, and zero otherwise. Afterwards, we can learn user (concepts) representations for each meta-path using GCNs.

Graph Convolution Networks (GCNs). GCNs learn node representations of a graph by inspecting neighboring nodes. In this work, we adopt the following layer-wise propagation rule to learn user (concept) representations/embeddings with respect to a meta-path.

$$\mathbf{h}^{(l+1)} = g(\mathbf{P}\mathbf{h}^l\mathbf{W}^l) \tag{1}$$

where $g(\cdot)$ is an activation function which we use $ReLu$ [25] here. $\mathbf{P} = \mathbf{D}^{-1}\widetilde{\mathbf{A}}$ where $\mathbf{D}$ is the diagonal node degree matrix of $\mathbf{A}$ to normalize the matrix $\widetilde{\mathbf{A}}$, and $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is an adjacency matrix with self-loops in a graph based on a specific meta-path. $\mathbf{W}^l$ refers to a trainable weight matrix at layer l for all nodes. $\mathbf{h}^0$ can be fed with features of each node if there is a set of features for each node or can be initialized and learned afterwards as well. The output representation of the last layer can be used as user (concept) representations. For example, when $l = 2$, the representation of a user u for a meta-path MP_i will be $\mathbf{e}_u^{MP_i} = \mathbf{h}^3_{u_{MP_i}}$ where $\mathbf{h}^3_{u_{MP_i}}$ is the output of the last layer of GCNs for the meta-path MP_i with respect to u. In our study, we use a single layer GCN where $\mathbf{h}^0$ is initialized randomly and learned during the training process, but one can easily extend it with more

layers or using existing features for $\mathbf{h}^0$.

Attention. Attention mechanism [36] is motivated by how we pay visual attention to different regions of an image or relevant words in one sentence, and has been used widely to advance various fields such as natural language processing and recommender systems [37, 13]. In our context, different meta-paths can have different importance with respect to each user, and incorporating the importance of each meta-path differently for each user can be beneficial when aggregating user representations from different meta-paths, i.e., $\{\mathbf{e}_u^{MP_1} \dots \mathbf{e}_u^{MP_{|MP|}}\} \rightarrow \mathbf{e}_u$. In our work, we apply the attention mechanism from [6] for our context as follows:

$$\alpha_u^{MP_i} = \frac{exp(\mathbf{V}_u^T \sigma(\mathbf{W}_u \mathbf{e}_u^{MP_i}))}{\sum_{j \in |MP|} exp(\mathbf{V}_u^T \sigma(\mathbf{W}_u \mathbf{e}_u^{MP_j}))} \tag{2}$$

where the output $\alpha_u^{MP_i}$ indicates the weight (or importance) for $\mathbf{e}_u^{MP_i}$, and $\mathbf{V}_u^T$ and $\mathbf{W}_u$ are trainable matrices for users. The attention mechanism can be formulated in the same manner for concepts. Next, the user representations coming from different meta-paths can be aggregated as follows:

$$\mathbf{e}_u = \sum_{j \in |MP|} \alpha_u^{MP_j} \mathbf{e}_u^{MP_j} \tag{3}$$

The above-mentioned attention mechanism takes into account different meta-paths but does not consider any context in the extended MF, which can be the latent features of users and concepts for MF. Therefore, we also investigate the following attention mechanism which considers the latent features of user $\mathbf{x}_u$, which has not been explored in previous studies. In this case, a meta-path based embedding $\mathbf{e}_u^{MP_i}$ and $\mathbf{x}_u$ are concatenated together when calculating the attention scores as follows.

$$\beta_u^{MP_i} = \frac{exp(\mathbf{V}_u^T \sigma(\mathbf{W}_u [\mathbf{e}_u^{MP_i}; f(\mathbf{x_u})]))}{\sum_{j \in |MP|} exp(\mathbf{V}_u^T \sigma(\mathbf{W}_u [\mathbf{e}_u^{MP_j}; f(\mathbf{x_u})]))} \tag{4}$$

where $f(\mathbf{x_u})$ applies non-linearity with a single layer feed-forward neural networks to $\mathbf{x_u}$ instead of using it directly, which is inspired by [31] where the authors showed that non-linear fusion is required when combining latent features from matrix factorization and entity embeddings from GCNs. Afterwards, the final user representation can be obtained in the same manner as Eq. 3.

$$\mathbf{e}_u = \sum_{j \in |MP|} \beta_u^{MP_j} \mathbf{e}_u^{MP_j} \tag{5}$$

The attention mechanism can be formulated in the same manner for concepts.

Prediction. Given those learned user and concept representations/embeddings $\mathbf{e}_u$ and $\mathbf{e}_c$. The preference score of a concept c for a user u can be calculated as follows by extending the matrix factorization framework:

$$\hat{y}_{u,c} = \mathbf{x}_u^T \mathbf{z}_c + \gamma \cdot \mathbf{e}_u^T \mathbf{M} \mathbf{e}_c + b_c \tag{6}$$

where $\hat{y}_{u,c}$ is the preference score, $\mathbf{x}_u$ and $\mathbf{z}_c$ are the latent features for the matrix factorization, and b_c is a bias term. In addition, $\mathbf{M}$ is a trainable matrix to let $\mathbf{e}_u$ in the same space with $\mathbf{e}_c$, and γ is a trainable parameter for the trade-off between the prediction scores from matrix factorization and the user and concept embeddings.

4.1 Training Details

Loss function. We use the Bayesian Personalized Ranking (BPR) [29] which has been widely used for recommender systems with implicit feedback [7, 5, 27]. The intuition behind BPR is that a learned concept for a user should be ranked higher (with a higher score) compared to a random one in the list of concepts with which the user has not interacted, which can be formulated as follows:

$$\mathcal{L} = \sum_{(u,i,j) \in D_s} -ln(\sigma(\hat{y}_{uij})) + \lambda \left\| \Theta \right\|^2 \tag{7}$$

where (u, i, j) refers to a triplet including a user u, an interacted concept i and an unknown concept j for the user. $\hat{y}_{uij} = \hat{y}_{ui} - \hat{y}_{uj}$ measures the preference difference between the interacted concept and the unknown one, σ denotes the *sigmoid* function: $s(x) = \frac{1}{1+e^{-x}}$, λ is the regularization parameter for the $\mathcal{L}_2$ norm, and Θ denotes the set of parameters to be learned. The training set D_s can be constructed by paring an unknown concept randomly with an interacted concept in the training set of a user.

To learn the parameters of our proposed approach for minimizing the loss in Eq. 7, we use a mini-batch gradient decent with 1,024 as the batch size, and use the Adam update rule [20] to train the model using the training set. In addition, the learning rate is set as 0.01, the regularization parameter λ is set as $1e-8$, and the dimension of latent features for MF and that of user (concept) embeddings are set as 30 and 100 respectively as in [13].

To overcome the overfitting problem, we further construct a validation set by using the last interacted concept for each user, and randomly pair each known concept with 99 unknown concepts. We run 500 epochs where the convergence is observed, and monitor the performance of evaluation metrics (see Section 5) on the validation set. At the end, we choose the best-performing model on the validation set in terms of *MRR* (Mean Reciprocal Rank), which is one of the evaluation metrics measuring how well a ground truth concept is ranked in the corresponding set of 100 concepts. Any other evaluation metric can be used for choosing the best-performing model as well based on the preference for a specific metric.

5. EXPERIMENTAL SETUP

MOOC Dataset. We use the MOOCCube dataset [39] from the XuetangX platform for our experiments. The MOOCCube dataset is one of the largest and comprehensive MOOC datasets, and provides rich information about MOOCs and user activities on the platform from 2017 to 2019 [39]. Each course or video has a set of covered knowledge concepts in the dataset. In this work, we use user activities from 2017-01-01 to 2019-10-31 for training and those from 2019-11-01 to 2019-12-31 for testing. We limit users who have learned concepts in both training and testing periods and have at least one new concept (which did not appear in the training

Table 2: Statistics of the MOOCCube dataset for experiments.

Entities	Statistics	Relations	Statistics
users	2,005	user-concept	930,553
concepts	21,037	user-course	13,696
courses	600	course-video	42,117
videos	22,403	teacher-course	1,875
schools	137	video-concept	295,475
teachers	138	course-concept	150,811

period) in the testing period. Overall, the dataset consists of 2,005 users 21,037 concepts, 600 courses, 22,403 videos, 137 schools, 138 teachers, and the relationships among those entities. In total there are 930,553 interactions between users and concepts with 858,072 interactions in the training set and the rest (72,481) in the test set. The overall statistics of the dataset are presented in Table 2.

Evaluation Metrics. We evaluate the top–k predictions of concepts for users with the following widely used evaluation metrics where k is set to 5, 10, and 20. We calculate all metrics for each set of 100 concepts (with one interacted and 99 unknown) in the test set. For each interacted concept with respect to a user u, we generate the corresponding recommendation list $R_u = \{r_u^1, r_u^2, \ldots, r_u^k\}$ where r_u^i indicates concept ranked at the i–th position in R_u based on the predicted scores of those concepts.

Hit Ratio of top–k concepts ($HR@k$) measures the fraction of relevant concepts in the test set that are in the top–k concepts of the recommendations: $HR@k = \frac{1}{N}\sum_u I(|R_u \cap T_u|)$ where N is the total number of sets for testing, $I(x)$ is an indicator function which equals one if $x > 0$ and equals zero otherwise. *Normalized Discounted Cumulative Gain* ($nDCG@k$) takes into account rank positions of the relevant concepts, and can be computed as follows:$nDCG@k = \frac{1}{Z}DCG@k = \frac{1}{Z}\sum_{j=1}^{k}\frac{2^{I(|r_u^j \cap T_u|)}-1}{\log_2(j+1)}$ where Z denotes the score obtained by an ideal top–k ranking which serves as a normalization factor. *Mean Reciprocal Rank* (MRR) is the average of the reciprocal ranks of positive concepts: $MRR = \frac{1}{N}\sum_1^N \frac{1}{rank_i}$ where $rank_i$ refers to the rank position of the one interacted concept in the corresponding set of 100 concepts with the rest of unknown ones.

We use the paired *t-test* for testing the significance where the significance level of α is set to 0.05 unless otherwise noted.

5.1 Compared Methods

To better understand and investigate the contribution of each component and the performance with the two attention mechanisms introduced in Section 4, we first compare several variants of our approach. $\texttt{MOOCIR}_{\text{a1}}$ denotes our approach with the attention mechanism only considering different meta-paths using Eq. 3. $\texttt{MOOCIR}_{\text{a2}}$ refers to our approach with the attention mechanism incorporating the latent features of users (concepts) using Eq. 4. $\texttt{MOOCIR}_{\text{a-}}$ is a variant of our approach without any attention, i.e., different meta-paths are treated *equally* and the representations learned from those paths are averaged. $\texttt{MOOCIR}_{\text{mf-}}$ refers to a variant without the matrix factorization part for prediction in Eq. 6, which only uses meta-path-based user and concept representations for predicting the preference score of a concept.

Next, we compare `MOOCIR` with the following baselines and state-of-the-art methods to evaluate the performance of recommending knowledge concepts for users. `TopPop` is a straightforward baseline method which ranks concepts based on their popularity. Here, the popularity of a concept can be measured based on the number of users that have learned the concept. `MFBPR` [29] is a matrix factorization approach which optimizes a pairwise ranking loss for the recommendation task as our approach but without meta-path-based representation learning. That is, the second component in Eq. 6 based on user (concept) representations is removed. `FISM` [17] is an item-to-item collaborative filtering approach which provides recommendations based on the average embeddings of all interacted concepts and the embeddings of the target concept. `NAIS` [15] is also an item-to-item collaborative filtering approach, but with an attention mechanism, which is capable of distinguishing which historical items in a user profile are more important for a prediction. We use the author's implementation for both `NAIS` and `FISM`[5]. `metapath2vec` [11]. `metapath2vec` is a meta-path-based representation learning model which leverages meta-path-based random walks to construct the heterogeneous neighborhood of a node and then leverages a heterogeneous skip-gram model to learn node embeddings. We use the StellarGraph [10] implementation of `metapath2vec` for our experiment in which the parameters of `metapath2vec` are set the same as in [11] except the number of random walks is set as 500 instead of 1000[6]. `ACKRec` [13] also models the MOOC dataset as a HIN and extracts user (concept) representations from the same set of meta-paths in Table 1. However, `ACKRec` treats the problem as rating prediction task where the rating of a concept for a user is the number of interactions between the user and the concept. Also, it exploits user and concept representations as features while extending the matrix factorization framework. We use the author's implementation[7] for our experiments. `MFBPR` and those `MOOCIR` variants are implemented using Tensorflow [1]. All experiments are run on an Intel(R) Core(TM) i5-8365U processor laptop with 16GB RAM, and `MOOCIR` variants take less than two days for training.

6. RESULTS

Table 3 summarizes the results using the variants of `MOOCIR`. As we can see from the table, $\texttt{MOOCIR}_{\text{mf-}}$ — which uses user and concept representations learned based on meta-paths with the HIN but without the matrix factorization component — provides worse performance compared to the other variants. The results indicate that extending the matrix factorization is necessary for `MOOCIR`.

Next, we compare $\texttt{MOOCIR}_{\text{a-}}$ and the variants with attention mechanisms (i.e., $\texttt{MOOCIR}_{\text{a1}}$ and $\texttt{MOOCIR}_{\text{a2}}$). We observe that both $\texttt{MOOCIR}_{\text{a1}}$ and $\texttt{MOOCIR}_{\text{a2}}$ outperform $\texttt{MOOCIR}_{\text{a-}}$ in terms of all evaluation metrics, which shows that using at-

[5] `https://github.com/AaronHeee/Neural-Attentive-Item-Similarity-Model`

[6] We noticed that using 1000 random walks took more than 10 days for training and did not improve the performance compared to using 500.

[7] `https://github.com/JockWang/ACKRec`

Table 3: Performance of several variants of our proposed approach in term of different evaluation metrics with the best-performing scores in bold.

	HR			*nDCG*			*MRR*
	k=5	10	20	k=5	10	20	
MOOCIR$_{mf^-}$	0.676	0.812	0.906	0.499	0.543	0.567	0.468
MOOCIR$_{a^-}$	0.701	0.832	0.920	0.513	0.556	0.578	0.477
MOOCIR$_{a1}$	**0.704**	0.836	**0.922**	**0.520**	**0.562**	**0.584**	**0.484**
MOOCIR$_{a2}$	0.703	**0.838**	**0.922**	0.517	0.561	0.583	0.482

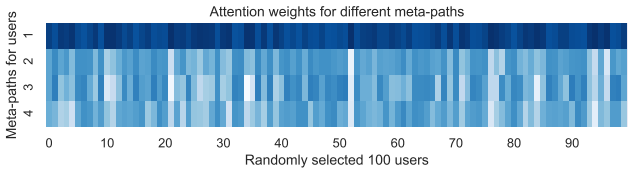


Figure 3: Attention weights of different meta-paths for 100 randomly chosen users learned using MOOCIR$_{a1}$ where we observe different weights of meta-paths for each user. In this heatmap, a darker cell indicates a higher attention weight.

tention can indeed improve the performance and different meta-paths have different importance for deriving user (concept) representations. This can be verified further by investigating learned attention weights for different meta-paths in MOOCIR as well. For example, Fig. 3 shows a heatmap regarding the learned attention weights for 100 randomly selected users using MOOCIR. In the figure, x-axis refers to the 100 users and y-axis indicates the attention weights for the four different meta-paths for users described in Table 1 in Section 4. From the figure, we can notice that the first meta-path (i.e., $user \rightarrow concept \xrightarrow{-1} user$) overall has a higher weight compared to others. In addition, we observe that the attention weights vary across users, which indicates the importance of each meta-path varies for different users.

Finally, by comparing the two different attention mechanisms, we observe that the one incorporating the latent features of users and concepts (Eq. 4) does not improve the performance compared to the simpler one (Eq. 3), which is different from our assumption. Instead, we observe that MOOCIR$_{a2}$ performs significantly worse than MOOCIR$_{a1}$ in terms of *HR@10* and *HR@20* for the users who have interacted with a limited number of concepts. Table 4 shows the performance for three groups of users with less than 150, 350, and 550 concepts, respectively. As we can see form the figure, MOOCIR$_{a1}$ outperforms MOOCIR$_{a2}$ significantly for the first group of 353 users. The results suggest that fusing information from the latent features of users (concepts) into the attention mechanism is a non-trivial task, and other approaches should be investigated in the future.

Table 4: Results of *HR@10* and *HR@20* for MOOCIR$_{a1}$ and MOOCIR$_{a2}$ for three groups of users (G150, G350, G550) with less than 150, 350, 550 concepts in the training set.

	HR@10			*HR@20*		
	G150	G350	G550	G150	G350	G550
MOOCIR$_{a1}$	0.806	0.830	0.851	0.894	0.911	0.927
MOOCIR$_{a2}$	0.801	0.829	0.852	0.886	0.908	0.925

Table 5: Performance of MOOCIR$_{a1}$ and compared methods in term of different evaluation metrics with the best-performing scores in bold.

	HR			*nDCG*			*MRR*
	k=5	10	20	k=5	10	20	
TopPop	0.486	0.629	0.767	0.343	0.390	0.425	0.332
MFBPR	0.668	0.811	0.907	0.481	0.527	0.552	0.448
FISM	0.584	0.701	0.800	0.438	0.476	0.501	0.418
NAIS	0.568	0.691	0.811	0.420	0.461	0.491	0.403
metapath2vec	0.642	0.773	0.873	0.468	0.511	0.537	0.440
ACKRec	0.659	0.764	0.842	0.503	0.538	0.557	0.475
MOOCIR$_{a1}$	**0.704**	**0.836**	**0.922**	**0.520**	**0.562**	**0.584**	**0.484**

Overall, MOOCIR$_{a1}$ provides the best performance among all variants. In the following, we discuss the performance of MOOCIR$_{a1}$ compared with other baselines and state-of-the-art methods.

Table 5 shows the performance of MOOCIR$_{a1}$ and compared methods. We first observe that all the other methods outperform TopPop which is a baseline method recommending popular concepts. For example, MOOCIR$_{a1}$ and ACKRec improves *MRR* over TopPop 45.8% and 43.1%, respectively. Among all the compared methods in Table 5, MOOCIR$_{a1}$ provides the best performance followed by ACKRec, MFBPR, and metapath2vec. ACKRec performs best in terms of *nDCG* and *MRR*, and MFBPR performs best in terms or *HR* among compared methods. In detail, a significant improvement of MOOCIR$_{a1}$ over ACKRec in *MRR* (+1.9%), *nDCG@5* (+3.1%), *nDCG@10* (+4.5%), *+nDCG@20* (4.8%) can be noticed ($\alpha < 0.01$). Compared to MFBPR, MOOCIR$_{a1}$ improves the *HR* scores 6.7%, 9.2%, and 9.4% when k =5, 10, 20, respectively ($\alpha < 0.01$). The two item-item CF methods (FISM and NAIS) do not perform well compared to MFBPR and ACKRec. One possible explanation might be due to the sparsity of the dataset, which makes that deriving item-item similarities based on interacted users for each item is challenging and limits the performance.

Those results indicate that the proposed approach MOOCIR$_{a1}$ can achieve competitive performance in terms of those evaluation metrics for top–k concept recommendations compared to the baselines and state-of-the-art methods.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we presented MOOCIR for predicting and recommending concepts that might be of users' interest on MOOC platforms. The comparison of MOOCIR variants in Section 6 shows that extending the matrix factorization with user and concept representations learned from different meta-paths and using attention for deriving those representations play crucial roles in achieving better performance. In addition, the results compared to other baselines and state-of-the-art methods indicate that MOOCIR$_{a1}$ can improve the performance of predicting and recommending concepts significantly. The comparison between the two introduced attention mechanisms (Eq. 3 and 4) suggests that a more comprehensive approach is required while fusing the latent features of users and concepts into the attention mechanism, which will be investigated in the near future.

8. REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283, 2016.

[2] K. Abhinav, V. Subramanian, A. Dubey, P. Bhat, and A. D. Venkat. Lecore: A framework for modeling learner's preference. In *EDM*, 2018.

[3] A. Agrawal, J. Venkatraman, S. Leonard, and A. Paepcke. Youedu: addressing confusion in mooc discussion forums by recommending instructional video clips. 2015.

[4] F. ALSaad and A. Alawini. Unsupervised approach for modeling content structures of moocs. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, pages 18–28, 2020.

[5] V. W. Anelli, T. Di Noia, E. Di Sciascio, A. Ragone, and J. Trotta. How to make latent factors interpretable by feeding factorization machines with knowledge graphs. In *International Semantic Web Conference*, pages 38–56. Springer, 2019.

[6] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Y. Cao, X. Wang, X. He, Z. Hu, and T.-S. Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161, 2019.

[8] Z. Chen, A. Brandon, C. Gayle, E. Nicholas, K. Daphne, and J. E. Ezekiel. Who's benefiting from moocs, and why. *Harvard Business Review*.

[9] Y. Dai, Y. Asano, and M. Yoshikawa. Course content analysis: An initiative step toward learning object recommendation systems for mooc learners. *International Educational Data Mining Society*, 2016.

[10] C. Data61. Stellargraph machine learning library. `https://github.com/stellargraph/stellargraph`, 2018.

[11] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017.

[12] J. Gardner, Y. Yang, R. S. Baker, and C. Brooks. Modeling and experimental design for mooc dropout prediction: A replication perspective. *International Educational Data Mining Society*, 2019.

[13] J. Gong, S. Wang, J. Wang, W. Feng, H. Peng, J. Tang, and P. S. Yu. Attentional Graph Convolutional Networks for Knowledge Concept Recommendation in MOOCs in a Heterogeneous View. In *Proceedings of the 43rd SIGIR Conference on Research and Development in Information Retrieval*, pages 79–88, 2020.

[14] H. Hajri, Y. Bourda, and F. Popineau. Personalized recommendation of open educational resources in moocs. In *International Conference on Computer Supported Education*, pages 166–190. Springer, 2018.

[15] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2354–2366, 2018.

[16] X. Jing and J. Tang. Guess you like: course recommendation in moocs. In *Proceedings of the International Conference on Web Intelligence*, pages 783–789, 2017.

[17] S. Kabbur, X. Ning, and G. Karypis. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD*, pages 659–667, 2013.

[18] A. Khalid, K. Lundqvist, and A. Yates. Recommender systems for moocs: A systematic literature survey (january 1, 2012 – july 12, 2019). *The International Review of Research in Open and Distributed Learning*, 21(4):255–291, Jun. 2020.

[19] H. Khalil and M. Ebner. Moocs completion rates and possible methods to improve retention-a literature review. In *EdMedia+ innovate learning*, pages 1305–1313. Association for the Advancement of Computing in Education (AACE), 2014.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] H. Labarthe, F. Bouchet, R. Bachelet, and K. Yacef. Does a peer recommender foster students' engagement in moocs?. *International Educational Data Mining Society*, 2016.

[22] X. Li, T. Wang, H. Wang, and J. Tang. Understanding user interests acquisition in personalized online course recommendation. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 230–242. Springer, 2018.

[23] F. Mi and B. Faltings. Adaptive sequential recommendation for discussion forums on moocs using context trees. In *Proceedings of the 10th international conference on educational data mining*, number CONF, 2017.

[24] C. Milligan and A. Littlejohn. Why study on a mooc? the motives of students and professionals. *International Review of Research in Open and Distributed Learning*, 18(2):92–102, 2017.

[25] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[26] Y. Pang, C. Liao, W. Tan, Y. Wu, and C. Zhou. Recommendation for mooc with learner neighbors and learning series. In *International Conference on Web Information Systems Engineering*, pages 379–394. Springer, 2018.

[27] G. Piao and J. G. Breslin. Transfer learning for item recommendations and knowledge graph completion in item related domains via a co-factorization model. In *European Semantic Web Conference*, pages 496–511. Springer, 2018.

[28] B. Prenkaj, P. Velardi, D. Distante, and S. Faralli. A reproducibility study of deep and surface machine learning methods for human-related trajectory prediction. In *Proceedings of the 29th ACM CIKM*, pages 2169–2172, 2020.

[29] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint*

arXiv:1205.2618, 2012.

[30] D. T. Seaton, Y. Bergner, I. Chuang, P. Mitros, and D. E. Pritchard. Who does what in a massive open online course? *Communications of the ACM*, 57(4):58–65, 2014.

[31] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31(2):357–370, 2018.

[32] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *Proceedings of the 24th ACM CIKM*, pages 453–462, 2015.

[33] Y. Sun and J. Han. Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter*, 14(2):20–28, 2013.

[34] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011.

[35] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD*, pages 797–806, 2009.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[37] C. Wu, F. Wu, M. An, J. Huang, Y. Huang, and X. Xie. Npa: neural news recommendation with personalized attention. In *Proceedings of the 25th ACM SIGKDD*, pages 2576–2584, 2019.

[38] M. Yao, S. Sahebi, and R. F. Behnagh. Analyzing student procrastination in moocs: A multivariate hawkes approach. *International Educational Data Mining Society*, 2020.

[39] J. Yu, G. Luo, T. Xiao, Q. Zhong, Y. Wang, J. Luo, C. Wang, L. Hou, J. Li, and Z. Liu. MOOCCube: A Large-scale Data Repository for NLP Applications in MOOCs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3135–3142, 2020.

[40] X. Yu, X. Ren, Q. Gu, Y. Sun, and J. Han. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI HINA*, 27, 2013.

[41] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 283–292, 2014.

[42] F. Zarrinkalam, S. Faralli, G. Piao, E. Bagheri, et al. Extracting, mining and predicting users' interests from social media. *Foundations and Trends® in Information Retrieval*, 14(5):445–617, 2020.

[43] J. Zhao, C. Bhatt, M. Cooper, and D. A. Shamma. Flexible learning with semantic visual exploration and sequence-based recommendation of mooc videos. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

Automatic Assessment of the Design Quality of Python Programs with Personalized Feedback

J. Walker Orr
George Fox University
jorr@georgefox.edu

Nathaniel Russell
George Fox University
nrussell18@georgefox.edu

ABSTRACT

The assessment of program functionality can generally be accomplished with straight-forward unit tests. However, assessing the design quality of a program is a much more difficult and nuanced problem. Design quality is an important consideration since it affects the readability and maintainability of programs. Assessing design quality and giving personalized feedback is very time consuming task for instructors and teaching assistants. This limits the scale of giving personalized feedback to small class settings. Further, design quality is nuanced and is difficult to concisely express as a set of rules. For these reasons, we propose a neural network model to both automatically assess the design of a program and provide personalized feedback to guide students on how to make corrections. The model's effectiveness is evaluated on a corpus of student programs written in Python. The model has an accuracy rate from 83.67% to 94.27%, depending on the dataset, when predicting design scores as compared to historical instructor assessment. Finally, we present a study where students tried to improve the design of their programs based on the personalized feedback produced by the model. Students who participated in the study improved their program design scores by 19.58%.

Keywords

Assessment, neural networks, intelligent tutoring

1. INTRODUCTION

Recently there has a been a lot of work in the development of tools for education in programming and computer science. Specifically there are many systems for intelligent tutoring which are designed to help students learn how to solve a programming challenge. The tutoring involved is primarily focused in suggesting functional improvements, that is, how to finish the program so that it works correctly.

Intelligent tutors such as [4] uses reinforcement learning to predict a useful hint in the form of an edit to a student's program that will get them one step closer to the goal of a functioning program. It uses histories of edits made by students, starting with a blank slate and ultimately terminating with a functional program to train the model. The system is based on *Continuous Hint Factory* [9] which uses a regression function to predict a vector that represents the best hint then translates that vector into a human-readable edit. Similarly [11] used a neural network to embed programs and predict the program output. Using that model of the program output, an algorithm was developed to provide feedback to the student on how to correct their program. Also, [16] use a recurrent neural network to predict student success at a task given a history of student submissions of their program for evaluations.

All these systems model student programs from *Hour of Code* [2]. *Hour of Code* is a massively open online course platform that teaches people how to code with a visual programming language. The language is simple and does not contain control constructs such as *loops*.

Moreover, the combination of language and problem setting are simple enough that there is a single or very few functional solutions for each problem [4]. This level of simplicity precludes the consideration of program design. However for general purpose programming languages such as Python, there are many ways of creating functionally equivalent programs. It is important for the sake of maintainability, modularity, clarity, and re-usability that students learn how to design programs well.

When it comes to the quality of design, there are varying standards. Further, some standards are more objective or easier to precisely identify that others. For example, the use of global variables are both widely recognized as poor design and are easy to identify. For some programming languages, "linters" exist to apply rules to check for common design flaws. For Python, Pylint [12] is a code analysis tool to detect common violations of good software design. It detects design problems such as the use of global variables, functions that are too long or take too many arguments, and functions that use too many variables. Pylint is design to enforce the official standards of the Python programming community codified in PEP 8 [15].

There are aspects of good design that are difficult to identify. For example, simple logic is a good design idea, but is quite nebulous. The complexity of a program's logic is con-

Walker Orr and Nathaniel Russell "Automatic Assessment of the Design Quality of Python Programs with Personalized Feedback". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 495-501. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

textual, it entirely depends on the problem the program is solving. Also, modularity is universally judged as a quality of good design, however it is not always clear to what extent a program should be made modular. How many functions or classes are too many? Again, it depends on the context of the problem for which the program is designed.

In a professional setting, code reviews are often practiced to promote quality design that goes beyond the straightforward rules of "linters." Code reviews are a manual process which require a lot of human effort. A recently developed system call DeepCodeReviewer [5] automates the code review process with a deep learning model. By using proprietary data on historical code reviews taken from a Microsoft software version control system, DeepCodeReviewer was trained to successfully annotate segments of C# code with useful comments on the code's quality.

However, to our knowledge, there is no system to perform in-depth code analysis for the purposes of evaluating and assessing design for general purpose languages in an educational context. The process of assessing the design of a program is time consuming for instructors and teaching assistants and it is an important component of complete intelligent tutoring system. Such a system needs to be adjusted or calibrated for the context of particular problems or assignments since there are important aspects of software design are context dependent. Moreover the system needs to match the particular standards of an instructor. Hence we propose a system that models design quality with a neural network trained on previously assessed programs.

1.1 Our Approach and Contribution

We propose a design quality assessment system based on a feed-forward neural network that utilizes an abstract syntax tree (AST) to represent programs. The neural network is a regression model that is trained on assessed student programs to predict a score between zero and one. Each feature the model uses is designed to be meaningful to human interpretation and is based on statistics collected from the program's AST. We intentionally do not use deep learning as it would make the representation of the program difficult to understand. Personalized feedback is generated based on each feature of an individual program. By swapping a feature's value for an individual program with the average feature value of good programs, it is possible to determine which changes need to be made to the program to improve is design. The primary contributions of this work are the following:

- The first to explicitly predict the design quality of programs in an educational setting to the best of our knowledge.
- High efficacy with an accuracy from 83.67% to 94.27% with only small amounts of training data required.
- The first intelligent tutoring system for design quality for Python.
- Personalized feedback without the explicit training or annotation.

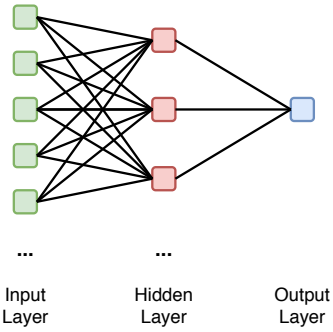


Figure 1: The model of program design quality, a feed-forward neural network. The "Input Layer" is the feature vector created from the AST. The "Hidden Layer" corresponds to calculation of x' specified in Equation 1. Finally, the "Output Layer" produces a single value, the design score as found in Equation 2.

2. METHOD

The task is to predict a design quality score for a student program written in Python. The score y is a real number between zero and one. The program is represented by a feature vector $\vec{x}$ produced by the output of a series of feature functions computed from the program's AST.

For an AST T, a series of feature functions $f_i((T)$ output is concatenated in to a feature vector $\vec{x}$ that represents key aspects of the program's design. The model $g(\vec{x}; \Theta)$ is a feed-forward neural network with a single hidden layer. It is a regression model that predicts the score y based on the feature vector $\vec{x}$ and parameters Θ.

2.1 Features

Despite recent advances in deep learning, we chose to represent the student program with feature functions computed on its AST. Deep learning is highly effective at learning useful feature representations of everything from images to time series to natural language texts. However, deep learning also requires large amounts of data and in this setting the quantity of manual annotated student programs is limited.

Additionally, AST are a natural and effective means of representing and understanding programs and can be created with free, available tools. An AST is an exact representation of the source code of program based on the programming language's grammatical structure. Producing an AST representation of a programming language is an essential first step in compilers and interpreters. The AST of a program contains all the content of its source but also is augmented with the syntactic relationships between every element. A parser and tokenizer to produce an AST for the Python programming language is provided by its own standard library. This makes the AST the natural representation to use, since it is free, convenient, exact, interpretable, and does not required any additional data. In contrast, deep learning would require a large amount of data to effectively reproduce the same representation.

Prior to representation as an AST, a program must first be broken into a series of tokens via the process of lexicaliza-

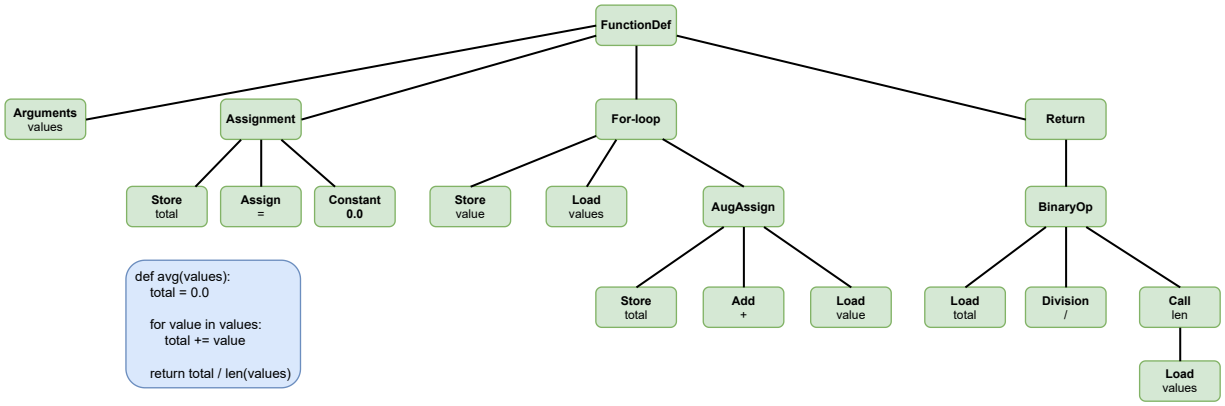


Figure 2: An abstract syntax tree for a segment of Python code that computes the average of the values in a list. The AST has been condensed for the sake of brevity and the restrictions of space. The related code segment is shown in blue.

tion. Lexicalization is the process of reading a program, character-by-character and dividing into work-like tokens. These tokens are also assigned a type such as a function call or variable reference. The grammatical rules of the language are applied to the lexicalized program to create the AST. In an AST, the leaf nodes of the tree are the program's tokens while the interior nodes correspond to syntactic elements and constructions. For example, an interior node could represent the body of a function or the assignment of a value to a variable. An example of an AST is found in Figure 2.

Given that an AST is a complete representation of a program, it is a natural basis for assessing the quality of a program's design. Deep learning may be able to automatically learning the same key syntactic relationships with enough data, however this information is simply available via AST. Further, features computed from the AST will be human interpretable unlike a representation produced by deep learning.

The features we created are all based on statistics collected from a program's AST. Some consist of simply counting the number of nodes of a given type, for example, the number of user defined functions. Other feature functions are based of subsections of the AST, such as the number of nodes per line or per function. Finally, some features are ratios or percentages such as the average percent of lines in a number in a function that are empty. All of the features are relatively simple and fast to compute, yet generally capture the design and quality of a program. Each of the feature functions $f_i(T)$ we defined can be found in Appendix A.

2.2 Model

The model is a feed-forward neural network [13] with a single hidden layer and single neuron in the output layer. The model's structure is illustrated in Figure 1. The values of the input layer are the feature vector $\vec{x}$. Each neuron in the hidden layer x'_j defined with the following equation:

$$x'_j = \text{ReLU}\left(\sum_{i=1}^{d} w_{i,j} x_i\right) \tag{1}$$

where d is the dimension of $\vec{x}$ and $w_{i,j} \in \Theta$ are the parameters, "weights" of the neuron. We use the ReLU [8] as the activation function for the hidden layer neurons. The final prediction of the design score is made by the output layer's single neuron:

$$y = \sigma\left(\sum_{j=1}^{d'} w_j x'_j\right) \tag{2}$$

where d' is the number of hidden layer neurons and $w_j \in \Theta$ are weights of this neuron. The function sigmoid is used because its domain spans from $(-\infty, \infty)$ but its range is $[0, 1]$ which ultimately guarantees the model always outputs a valid score. The model is trained with mean squared error as the loss function:

$$\text{MSE} = \frac{1}{n}\sum_{k=1}^{n}(y^*_k - y_k)^2 \tag{3}$$

where y^*_k is the ground-truth design score for the k^{th} instance i.e. program and n is the number of instances in the training data. The model is trained with the ADAM algorithm [7] and each parameter in the model was regularized according to their L2 norm [6]. For all our experiments, a hidden layer of size 32 was used. The model was trained for 250 epochs and the model from the best round according to a development set was selected for our experiments.

2.3 Ensemble

Due to the fact that fitting neural networks to data is a local optimization problem, the effect of initial values of the parameters Θ of the model remain after training. The process of training a neural network will produce a different model given the same data. This variation in the results of a trained model is particularly pronounced when the training data set is relatively small. To address this variation and mitigate its impact an ensemble of models can trained, each with different initial parameter values. Each model is independently trained and a single prediction is made by a simple of the average of individual predictions i.e.

$$y = \frac{1}{m}\sum_{l=1}^{m} y_l \tag{4}$$

where m is the number of models and y_l is the prediction of the l^{th} model. For our experiments, an ensemble of 10 models was used.

2.4 Personalized Feedback

The goal of intelligent tutors is to provide personalized feedback and suggestions on how to improve a program. The most straight-forward means of providing feedback would be to simply predict which possible improvements apply to a given program. However, training a model to directly predict relevant feedback would require a dataset of program with corresponding feedback and such a dataset can be hard to find or is expense to construct.

In order to avoid the need for a dataset with explicit feedback annotation, we use our model, trained on predicting design score, to evaluate how changes in program features would lead to a higher assessed score. Using the training data, we compute an average feature vector $\underline{\vec{x}}$ of all the "good" programs i.e. those with a design score greater than 0.75. To generate feedback for a program, its feature vector $\vec{x}$ is compared to the average $\underline{\vec{x}}$. For each feature, a new vector $\vec{x}'$ is created by replacing the feature value x_i with value with the average's value $\underline{x}_i$. This process is setting up a hypothesis, what if the program was closer to the average "good" program with regards to a particular feature? To answer this, the trained model $g(\vec{x};\vec{\theta})$ is used to predict a design score for the new vector i.e. $y_i' = g(\vec{x}';\vec{\theta})$. By comparing the original score of the program y with the new score y_i', the hypothesis can be tested. If the new score y_i' is greater than the original predicted score y, then the alteration of x_i to be closer to $\underline{x}_i$ is an improvement. Feedback based on this alteration is recommended to the student as personalized feedback. Since each feature in $\vec{x}$ is understandable to a human, feedback is given in the form of the suggestion to increase or decrease particular features. The suggestion for alteration is based on the comparison of x_i versus $\underline{x}_i$, if $x_i > \underline{x}_i$, the feedback of decrease x_i is given. In the other case, where $x_i < \underline{x}_i$ the feedback is to increase x_i. Based on the feature and the feedback of increase or decrease, a user-friendly sentence is selected from a table of predefined responses. For example, if x_i is the number of user defined functions and $x_i = 3$, $\underline{x} = 5$, and $y_i' > y$ then the feedback of "increase the number of user defined functions" is created.

3. EXPERIMENTS

The system was evaluated in two different experimental settings. The first evaluation is direct test of the model's accuracy on known design scores. For this, several different datasets and settings were compared against several baselines. The second evaluation is a small study of how student's responded to the system's feedback. Students were given feedback on the quality of their programs based on the model. They were given the chance to correct their programs after receiving feedback and have it manually reassessed.

3.1 Dataset

The dataset was collected over three years from an introduction to computer science course which teaches Python 3. It consists of four separate programming assignments which involve a wide range of programming skills. The simplest is "Travel," an assignment that involves the distance a vehicle travelled after going a constant speed for a specified duration. There are 118 student programs for "Travel." The next assignment, "Budget" is a budgeting program that lets a user specify a budget and expenses and determines if they are over or under their budget. 168 student programs were collected for "Budget". The third assignment in the dataset consists of creating a program to play "Rock-Paper-Scissors" against the computer. For this assignment, there are 111 student programs. The last assignment is programming the classic casino game "Craps" which involves rolling multiple dice and placing different types of bets and wagers. This assignment has 120 collected student programs.

All the assignments require the student to write the program from scratch in Python 3. The programs are to have a command-line, text-based interface and user validation. Students are required to use if-statements, loops, user defined functions. The "Craps" program also requires the student to do exception handling, and file I/O. A requirement of the program was to maintain a record of their winnings across sessions of playing the game, hence the results were required to be stored to a file. Also, the standards of design quality go up as the course progresses and since "Craps" is the last assignment, it has the highest standards. Each student program has an associated design score that was normalized to value between zero and one.

3.2 Baseline Methods

The model is compared against a variety of baseline regression methods. The simplest is linear regression, which simply learns a weight per each feature. Next is a regression decision tree which is trained with the CART algorithm [1]. It has the advantage over linear regression in that it can learn non-linear relationships. Non-linearity means a model can learn "sweet-spots" rather than simply having a "more is better" understanding of some features. For example, having some modularity in the form of user defined functions is good, however, too many is cumbersome. The "correct" number of user defined functions likely should fall into a relatively small range. Model selection on the maximum depth of the tree with a development set was used to determine that 10 was the best setting.

However, both of these models have the issue that they are not constrained to produce a score between zero and one, their prediction can be any real number. Hence another baseline method was used, created to be an intermediary step between linear regression and the neural network model. It is a linear model with a sigmoid transformation which guarantees the output be between zero and one. This model is effectively the final layer of the neural network model, i.e. the neural network without the hidden layer. The model is specified by the equation:

$$y = \sigma\left(\sum_{i=1}^{d} w_{i,j} x_i\right) \tag{5}$$

This model is also trained with ADAM [7]. All the baseline models and the neural network are trained with MSE as the loss function.

Method	Travel		Budget		RPS		Craps		Combined	
	MSE	Accuracy	MSE	Accuracy	MSE	Accuracy	MSE	Accuracy	MSE	Accuracy
Linear Regression	0.009	93.09%	0.032	87.43%	0.038	83.2%	**0.043**	**84.06%**	**0.027**	**87.03%**
Decision Tree	0.018	90.10%	0.031	87.26%	0.078	77.33%	0.072	79.41%	0.076	81.42%
Sig. Linear Regression	0.022	89.60%	0.046	85.13%	0.086	77.91%	0.063	81.43%	0.070	80.64%
Neural Network	0.007	93.48%	0.024	88.48%	0.041	83.9%	0.08	79.57%	0.033	85.61%
Ensemble	**0.005**	**94.27%**	**0.022**	**90.14%**	**0.022**	**87.66%**	0.053	83.67%	0.031	86.99%

Table 1: Design Score Prediction Results

3.3 Results

The model was compared versus each baseline in five different settings: the "Travel", "Budget", "Rock-Paper-Scissors", and "Craps" programs, and a combined dataset which includes all the programs. The results of the experiments can be found in Table 1. Each model is evaluated according to two different metrics: MSE and average accuracy. Average accuracy is defined as $\frac{1}{n}\sum_{k=1}^{n}\left(1-|y_k-y_k^*|\right)$.

Overall, the decision tree and the sigmoid-transformed were clearly the two worst models. This was surprising since decision trees are generally thought to be strictly more powerful than linear models. However, decision trees look for highly discriminative features to partition the data into more consistent groups. The under-performance of the decision tree possibly indicates that none of the features were especially indicative of a good or bad design on their own. Instead, the quality of a program is better described by a collection of subtle features, which gives credence to the belief that design quality is nuanced.

The reason sigmoid-transformed linear model under-performed linear regression was likely due to it being trained with ADAM. ADAM does not guarantee convergence to a global optimum like the analytical solution to linear regression. Apparently the restriction on predictions to be within the specified range of zero to one was not important.

Linear regression did surprisingly well, beating both the neural network and network ensemble in the "Craps" and combined datasets, though barely. In those cases, the difference between the ensemble and linear regression was less than a percent. This is likely due to the stability of linear regression's predictions. Though linear regression does not have the power and flexibility of neural networks this can also be a benefit by limiting how wrong their predictions are. Neural networks and even ensembles can make overconfident predictions on outliers or other unusual cases. The "Craps" dataset contained the most complex programs and it is likely a handful of predictions significantly brought down the average.

The network ensemble outperformed the single neural network in every case, which is to be expected. The margin of improvement of the ensemble versus the single neural network in accuracy on the four individual program datasets ranged from 1% to 4%. The network ensemble did the best overall by being the best in most cases or coming in a close second in all the other cases. The importance of using an ensemble is evident on the "Craps" dataset where the individual neural network under-performed significantly. On the other datasets, the neural network outperformed linear regression by a small margin, but on "Craps" the neural network model under-performed the linear regression model by 5%. Again, this is most likely due the instability and variability of neural network predictions i.e. small differences in features can lead to a large difference in the prediction. In the "Craps" dataset, the improvement of the ensemble over the single neural network model illustrates the relative stability of the ensemble's predictions. In every case, the ensemble is superior to the single neural network and had the best overall performance by producing the most accurate results on three of the datasets and effectively tying for the best on the other two.

One noticeable pattern was that all the models performed better on the "Travel" and "Budget" datasets than on the "RPS", "Craps", and combined datasets. Universally, the most difficult dataset was "Craps" which likely lowers the accuracy on the combined dataset. Due to the shifting standards and expectations of student assignments, a model per assignment appears to worthwhile. This is a bit counter-intuitive since there are many common standards and expectations across assignments.

Overall, the network ensemble produced reliable, accurate results when trained per dataset. The accuracy of the ensemble is arguably close to being useful in practical application. Further, comparing the scores of an instructor versus another instructor or even against themselves, the rate of agreement must be less than 100% and with an accuracy of the network ensemble ranging from 83.67% to 94.27%, the model's accuracy is possibly close to a realistic ceiling.

3.4 Feedback Study

In order the evaluate the effectiveness of the personalized feedback, we conducted a small study on the effect of the feedback on the design score of student programs. For the "Rock-Paper-Scissors" program the network ensemble was used to generate personalized feedback for the student programs instead of the usual instructor feedback. The network ensemble was the same as used in the design score experiments, it was trained with prior years worth of student programs. Having received the personalized feedback, students opted into correcting their program for extra credit on their assignment. The feedback was in form of a series of comments, where each comment was "increase" or "decrease" the name of a feature as described in Section 2.4.

The class is an introduction to computer science course with multiple sections and two different instructors. Students from both instructors participated in the study. Out of 73 students enrolled across the sections of the course, 15 stu-

dents chose to opt-in.

The revised programs were assessed again manually for design quality and the scores were compared against the originals. The design score of the programs started at an average of 68.33% and after the feedback and correction the average rose to 87.92%, a 19.58% absolute improvement. Using a paired t-test, the improvement was judged to be significant with a p-value of 0.001.

The results of the study suggest the feedback was generally useful in guiding students to improve the design quality of their programs. The improvement was noticeable to the instructors anecdotally as well. For example, the usage of global variables and "magic numbers" decreased significantly. Though the study does have some caveats including its small sample size and opt-in participation. It could be that those students willing to opt-in are those most willing or able improve with a second chance.

4. CONCLUSIONS & FUTURE WORK

Overall, we proposed a neural network model ensemble for predict the design quality score of a student program and experimentally demonstrated its effectiveness. Further, our system provided personalized feedback based on the difference between a program's feature values and the average features' value of "good" programs. A small study provides evidence that the feedback was of practical use to students. Students were able to improve their programs significantly based on the feedback they received.

There is also evidence that training models per assignment is most effective. However, the model needs to be evaluated on more programming assignments. Further, there is a possibility of utilizing transfer learning [10] to help the model learn what is in common across the assignments.

The feedback given was shown to be effective, but more nuanced feedback could be useful. Specifically, feedback targeted to individual lines or segments of code would possibly help students improve their program's more effectively. However, this may require additional supervision i.e. annotation for explicit training. Active learning [14] or multi-instance learning [3] may be alternatives to gathering additional annotation.

5. REFERENCES

[1] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[2] Code.org. Code.org: Learn computer science. https://code.org/research.

[3] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

[4] A. Efremov, A. Ghosh, and A. Singla. Zero-shot learning of hint policy via reinforcement learning and program synthesis. *International Educational Data Mining Society*, 2020.

[5] A. Gupta and N. Sundaresan. Intelligent code reviews using deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18) Deep Learning Day*, 2018.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[7] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[8] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[9] B. Paassen, B. Hammer, T. W. Price, T. Barnes, S. Gross, and N. Pinkwart. The continuous hint factory-providing hints in vast and sparsely populated edit distance spaces. *Journal of Educational Data Mining*, 2018.

[10] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[11] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas. Learning program embeddings to propagate feedback on student code. In *International conference on machine Learning*, pages 1093–1102. PMLR, 2015.

[12] Pylint.org. Pylint - code analysis for python. https://pylint.org.

[13] F. Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document, 1961.

[14] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[15] G. Van Rossum, B. Warsaw, and N. Coghlan. Pep 8. https://www.python.org/dev/peps/pep-0008/.

[16] L. Wang, A. Sy, L. Liu, and C. Piech. Learning to represent student knowledge on programming exercises using deep learning. *International Educational Data Mining Society*, 2017.

APPENDIX

A. FEATURE FUNCTIONS

- The number of functions
- The number of assignments
- AST nodes per function
- Lines of code per function
- Total lines of code
- Number of literals
- The proportion of white-space characters to the total number of characters
- Number of empty lines
- Deepest level of indentation
- Number of "if" statements
- Number of comments
- Number of AST nodes per lines of code
- Number of try-except statements

- AST nodes per try-except statement
- AST nodes per “if” statement
- Number of lists
- Number of tuples
- Average line number of literals
- Average line number of function definition
- Average line number of “if” statement
- Ratio of AST nodes inside functions versus total number of AST nodes
- Number of function calls
- Number of “pass” statements
- Number of “break” statements
- Number of “continue” statements
- Number of global variables
- Number of zero and one integer literals
- Average line number of “import” statement
- Number of numeric literals
- Number of comparisons
- Number of “return” statements
- Maximum number of “return” statements per function
- Maximum number of literals per “if” statement

Exploring the Importance of Factors Contributing to Dropouts in Higher Education Over Time

Hasan Tanvir, Irene-Angelica Chounta
University of Tartu, Estonia
{hasan.mohammed.tanvir, chounta}@ut.ee

ABSTRACT

The aim of this work is to provide data-driven insights regarding the factors behind dropouts in Higher Education and their impact over time. To this end, we analyzed students' data collected by a Higher Education Institute over the last 11 years and we explored how socio-economic and academic changes may have impacted student dropouts and how these changes may have been reflected or captured by students' data. To analyze the data, we engineered features that may predict student dropouts on three dimensions: academic background, students' performance and students' effort. Then we carried out a correlation analysis to investigate the potential relationship between these features and dropouts, we performed a multivariate analysis of variance (MANOVA) to investigate whether the engineered features change significantly among student cohorts with different admission year and, finally, we carried out a regression analysis to confirm that the engineered features' impact on predicting dropouts changes over the years. The results suggest that the importance of features regarding the academic background of students (such as the students' prior experience with the academic institution), and the effort students make (for example, the number of days students spend on academic leave) may change over time. On the contrary, performance-based features (such as credit points and grades) do interact with time suggesting that performance measures are stable predictors of dropouts over time. On the basis of the findings, we argue that the performance of prediction models for assessing students at risk of dropping out of their studies can be affected by the age of data and we outline the possibility of including a forgetting factor for non-recent data in order to leverage their impact on prediction performance.

Keywords

dropouts, feature engineering, predictive modeling, higher education

1. INTRODUCTION

Student retention is pivotal for success of an educational institute. To understand the reasons behind dropouts, individual cases of students had to be analyzed on one by one basis. The advent of information technology, the use of digital technologies by educational institutes and the collection of rich data regarding students' background, performance and effort offer the possibility of using advanced analytical approaches, such as machine-learning in order to identify trends and patterns that may indicate students at risk of dropping out from their studies [13].

To ensure quality education, Higher Education Institutes (HEIs) typically offer analytical solutions - such as, learning dashboards - to inform stakeholders (for example, program directors, academic specialists and instructors) with respect to student dropouts [2]. To do so, machine-learning models are typically employed to analyze data collected by Study Information Systems (SISs) and Learning Management Systems (LMSs) and to predict whether a student faces a risk to drop out from their studies [1, 4]. This is a well-established practice but little research has been carried out with respect to the temporal aspects of data, such as the age of data used to train predictive models for assessing dropouts. One may argue that – in terms of predictive performance – the more training data, the better. However, our hypothesis is that the factors affecting dropouts in Higher Education (HE) change significantly over time due to socio-economic conditions [16] and to such an extent that data age may affect the computational model's predictive accuracy.

The goal of this research is to analyze the data collected over 11 years, 2010 to 2020 from the SIS of a national European HEI. The objective is to engineer and identify the important log-based features behind dropouts, how these features may change over years, and to explore their impact on predicting dropouts. The contribution of this work is twofold:

- to provide insights regarding log-based features that may relate to student dropouts in HE;
- to explore the relationship between the aforementioned features and time regarding their impact on dropout prediction.

In the following section we provide a short overview of related research, then we present our methodological approach and we follow up with the results of our analysis. We conclude with a contextualized discussion on our findings, the

Hasan Tanvir and Irene-Angelica Chounta "Exploring the Importance of Factors Contributing to Dropouts in Higher Education Over Time". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 502-509. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

practical implications of this work and limitations as well as potential future directions.

2. RELATED WORK

As dropouts in Higher Education, we identify the cases of students who do not successfully complete their studies for reasons that indicate lack of motivation and willingness to pursue an academic degree. Dropouts in HE is a prominent issue with negative impacts for students, and institutions that also affects national and international policies[1].

The reasons behind students dropouts can vary between personal (for example, students feeling isolated or homesick [8]), academic (such as students' lack of background knowledge or study skills [9]) and socio-economical (for example, financial difficulties and cultural adaptation [16, 12]). At the same time, factors that relate to the academic institution rather than the students themselves (such as the quality of studies and resources that the institution offers [3]) can also affect student dropouts. Tinto's theoretical model of students' dropouts from college [15] identified two dimensions as crucial in terms of academic success: student's characteristics (such as family background and goals) and student's experience with the academic system (such as student's performance and relationship with mentors and colleagues). Crosling et al. [7] attributed student dropouts to the services the academic institutes offer to students, such as information regarding the admission process, quality of the teaching, and assessment and [11] investigated the relationship between the socio-economic status of a country and students dropouts. Other work [10] argued that student dropout is often related to a combination of reasons that include individual and curriculum-level factors, for example, inefficient study skills and inefficient academic or social environment.

In this work, we examine the case of an Estonian HEI. Estonia, being a relatively new member of European Union, is going through social, structural and economic changes in many sectors, including higher education. We argue that these socio-economic changes that arguably affect student dropouts, may also affect the performance of predictive algorithms that model student dropouts if temporal aspects of students' data (such as, the age of data as depicted for example by students' admission year) are not taken into account.

3. METHODOLOGY

This research was carried out in an Estonian Higher Education Institute (HEI). Recently, the HEI launched an initiative aiming to support students in successfully completing their studies. To do so, the HEI designed a learning analytics (LA) dashboard that provided information to academic stakeholders (in this case, program directors and academic specialists) regarding potential reasons that may contribute to dropouts in their programs and suggestions concerning appropriate feedback and support that they could offer to students-at-risk. To provide this information, the LA dashboard used students' data collected by the SIS of the HEI – with the students' informed consent – throughout the students' academic career [5].

To identify students at risk from dropping out from their studies, the LA dashboard used a predictive model (described in [5] that assessed dropout risk on three dimensions: academic background of the student, student's performance, and effort. The separation of the dimensions would help the institute to link dropout factors directly to students' cohorts. Each dimension was defined based on pre-selected engineered features from the SIS database. In this work, we used data collected for students on the bachelor level from 2010 to 2020) to explore whether the predictive features used by the model change over time, to what extent, and what is the impact of this change on dropout prediction.

3.1 Method of Study

Our hypothesis was that the performance of dropout predictive models that were trained with students data collected over various admission years, will not be consistent over time; the reason for that being that the predictive features change significantly over time. For the purpose of our research, we followed a three-step approach:

- we performed a correlation analysis to explore indications of potential relationships between log-based, engineered student features and dropouts per admission year;
- we carried out a MANOVA to establish that the log-based features retrieved from the correlation analysis vary significantly over student cohorts of different admission years;
- we performed a regression analysis with interaction terms to investigate the effect of the log-based features – retrieved from correlation analysis and MANOVA – on dropout prediction for student cohorts admitted on different years.

As a proof of concept, we trained a regression model as a binary classifier to predict student dropouts using the engineered features that we acquired from the aforementioned process. Then, we tested the performance of the classifier on unseen data. An overview of the method of study is presented in Figure 1.

3.2 Description of data

In this work, we used data of bachelor-level students that the HEI collected using the Study Information System (SIS) over a period of 11 years (from 2010 to 2020). The data was originally organized in 4 tables containing information regarding students' academic background, demographics, study place, and study info data.

In the SIS database, each student and each study place (or else, curriculum enrollment) have different unique identifiers ("person ID" and "study place ID", respectively). This consequently means that the relationship between students and curriculum enrollments is 1 to N - that is, one student may be enrolled in multiple curricula at the same time. In order to create one working dataset, we merged the four database

[1] http://publications.europa.eu/resource/cellar/d9de3b17-0dcf-11e6-ba9a-01aa75ed71a1.0001.01/DOC_1

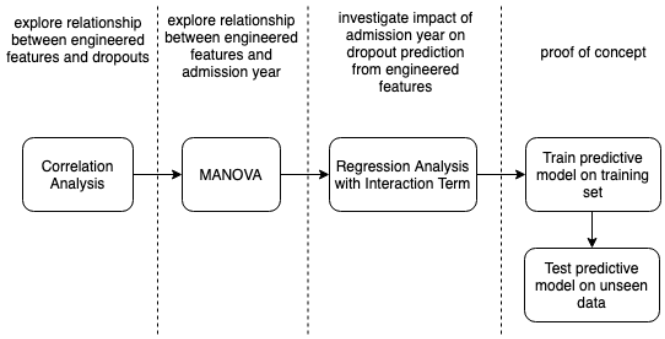


Figure 1: A graphical representation of the method of study, including the three-step analytical approach and the proof-of-concept example.

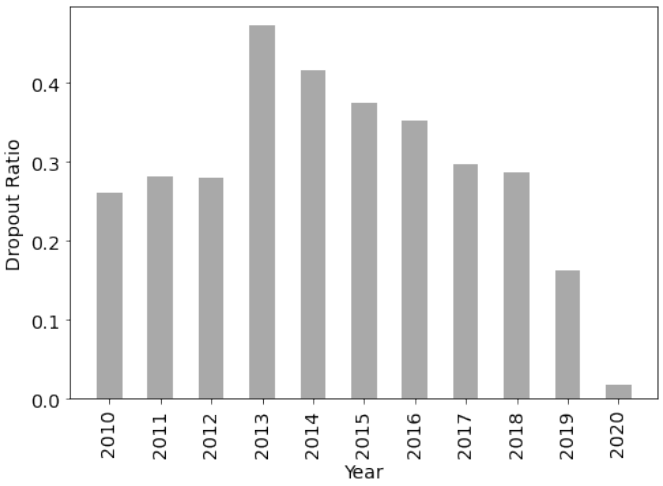


Figure 2: The dropout ratio, that is the ratio of the students who dropped out over the whole bachelor-level student population per admission year, from 2010 to 2020

tables using a combination of the unique keys "person ID" and "study place ID". Using this dataset, we engineered a set of features that can potentially describe student's academic profile on three dimensions – that is, student's academic background, student's academic performance, and student's academic effort – and may provide insights regarding students who may be at risk of dropping out from their studies. Following the recommendation of the ethics committee of the HEI, we excluded information that could be linked to students identity and demographic background to avoid potential discrimination, gender or racial bias.

As dropouts, we identified students who terminated their studies due to reasons (as recorded by the SIS of the HEI) that may indicate lack of motivation, or unwillingness to pursue an academic degree. Students can dropout at any point during the academic year, but the HEI records students' "exmatriculation" in the beginning of every semester. In total, the dataset consisted of 9623 students who are enrolled in the bachelor programs offered by the HEI. Out of these students, 3428 students dropped out at some point during their studies before they acquire an academic degree. Figure 2 shows the distribution of the dropout ratio – that is, the number of students who dropped out over the whole bachelor-level student population per admission year, over 11 years. For Year 2020 we only obtained data for the first academic semester (February to June).

3.3 Features Engineering

For each dimension of a student's academic career, we engineered a set of features from data recorded from the SIS of the HEI. In brief:

- Academic Background: The SIS records information regarding students' earlier academic background when students enroll to a study program offered by the HEI. We engineered features related to students earlier academic degrees, the admission score, admission special conditions (for example, good results in Olympiads, high scores in the academic aptitude test) and the number of previous enrollments to study programs offered by the same HEI.
- Performance: Here, we engineered features related to students' performance as depicted by grades and awarded credits throughout the study program. Performance-related features include credits earned, grades, and cumulative positive and negative study results (that is, numbers of passed and failed courses).
- Effort: Here, we considered features that can represent a student's overall effort during their studies. Some of these features are, the number of days a student spends on academic leave, the number of credits the student cancelled throughout the semester, the number of the registered courses during a semester and information about student's allowances and achievement stipends.

The complete set of features per dimension along with a short description for each is presented in the appendix (Table 4).

4. RESULTS

The results are presented per each step of the analytical process: the correlation analysis, the MANOVA and the regression analysis. For simplicity, we only report statistically significant findings at the $p < 0.05$ level. Then, we report our exploratory findings from the prediction example as proof-of-concept.

4.1 Correlation Analysis

We carried out a correlation analysis (Spearman's rank-order correlation) to explore the potential relationship between the engineered features and student dropouts. We only report statistically significant correlations at the $p < 0.05$ significance level with medium and strong correlation coefficients ($\rho \geq |0.3|$) (Table 1). The correlation analysis suggests that features representing student performance and effort, such as the number of credits a student earns or the number of courses they register, may relate negatively with the probability of dropping out from their studies (that is, the more courses they register, the less likely to dropout). One interesting finding was that the student's economic support was negatively correlated with student dropouts from 2010 to

	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Performance Features											
nr.of.courses.with.any.grade	-0.58	-0.65	-0.71	-0.59	-0.66	-0.72	-0.78	-0.71	-0.70	-0.35	
credits.earned	-0.72	-0.74	-0.74	-0.76	-0.79	-0.80	-0.80	-0.74	-0.73	-0.47	
extracurricular.credits.earned	-0.55	-0.56	-0.56	-0.53	-0.58	-0.54	-0.59	-0.43	-0.48		
all.results	-0.49	-0.59	-0.62	-0.53	-0.60	-0.66	-0.75	-0.68	-0.63	-0.36	
negative.results		0.32	0.38	0.37		0.30				0.31	
grade.A	-0.34	-0.42	-0.46	-0.46	-0.43	-0.45	-0.38	-0.34	-0.35		
grade.B	-0.47	-0.48	-0.50	-0.54	-0.56	-0.55	-0.53	-0.38	-0.41		
grade.C	-0.33	-0.35	-0.30	-0.40	-0.43	-0.46	-0.48	-0.35	-0.36		
grade.F	0.31										
passed	-0.66	-0.69	-0.65	-0.60	-0.58	-0.64	-0.65	-0.52	-0.35		
not.present				0.31							
Effort Features											
days.on.academic.leave			0.31								
days.studying.abroad	-0.30										
credits.cancelled		-0.34	-0.44								
nr.of.courses.registered	-0.58	-0.65	-0.71	-0.59	-0.66	-0.72	-0.78	-0.71	-0.74	-0.43	
credits.registered	-0.67	-0.71	-0.73	-0.57	-0.67	-0.71	-0.77	-0.73	-0.74	-0.47	
total_economic_support	-0.48	-0.58	-0.52	-0.31					-0.47		
study_period_in_years	-0.40	-0.63	-0.40	-0.40	-0.50	-0.59	-0.83	-0.72	-0.96	-0.87	

Table 1: Spearman's Rank Correlation for the engineered features and student dropouts per admission year. Here we present correlations where $\rho \geq |0.3|$ and $p < 0.05$. The features for the dimension of Academic Background did not appear to correlate strongly with dropouts over the admission years.

2013 but no correlation appears for the past few years (with the exception of 2018). This may suggest that presently students are in a better financial situation and can therefore afford studying until they complete their degrees. Alternatively, it may indicate a change in the state's or the university's policy regarding tuition fees.

The correlation analysis did not reveal any strong and significant relationship between dropouts and features of the student's academic background. However, correlations only suggest the potential existence of relationships. Therefore additional analysis is necessary to establish whether the importance of the engineered features on dropouts may change over time.

4.2 Multivariate Analysis of Variance

Next, we performed a one-way MANOVA to investigate whether the engineered features vary significantly for student cohorts admitted over different academic years. The engineered features were the dependent variables and the admission year was the independent variable for each of the dimensions. The results of the MANOVA are presented in Table 2. For the academic background dimension, all the features appear to be significantly different among the independent groups ($p < 0.05$) which may indicate that the academic background features are year-dependent. For both the performance and effort dimensions, the majority of features vary significantly among student cohorts of different admission years with $p < 0.05$.

Based on the MANOVA results we assume that the engineered features appear to be significantly different for student cohorts based on the admission year. This may consequently signify that the impact of the log-based features on dropout can be time-dependent.

4.3 Regression Analysis

To further explore whether the performance of a predictive model depends on temporal aspects of training data, we carried out a (logistic) regression analysis with the variable "*dropout*" as the dependent variable, the predictive features as the independent variables and admission year as the interaction term. Table 2 presents the features that interacted with admission year. Regarding students' academic background, we found that the students' previous experience with the HEI is dependent on admission year while the normalized admission score is significant in terms of regression analysis but marginal ($p = 0.07$) in terms of interaction with admission year. Time-dependency of previous experience with the HEI may reflect structural or policy changes of the academic institution that affect students' experience. Regarding the admission special conditions, we did not find any interaction with admission year or dropout (also evident from the correlation analysis). Furthermore, the results suggested that the students' previous study level – in case of master's level studies – may be important for dropout prediction and interact with admission year. A potential explanation could be that there is a confounding effect between the feature indicating previous studies in the same institution and the feature indicating previous study level.

Concerning students' performance, the results suggest that features such as the credits a student earns or the grades they are awarded can be used to indicate dropout risk. However, performance-based features do not appear to interact with admission year. In other words, their impact on predicting student dropout does not depend on the year a student was admitted in the academic institution. Regarding the importance of features that denote effort, such as the time a student spends on an academic leave, and the number of registered credits, they seem to have a different effect on student dropouts depending on the year of admission. This means that the features' weight on dropout predic-

tion changes when coupled with the interaction term. This may indicate that the impact of these features on student dropouts is not consistent over time. We did not find any indications that effort-related features such as the credits a student cancels over the semester or the duration of studies (study period in years) depend on admission year.

4.4 Proof of Concept

To further explore the impact of time on modeling student dropouts using log-based student features, we split the data in two sets: a training and a test set. For the training set, we included all records of students admitted from 2010 to 2020, except those who were admitted on 2011 and on 2018 (both points representing instances close to the chronological beginning and the ending of our data collection). We used the training set to train a regression model, and we used the trained model as a binary classifier to predict student dropouts on the test set of unseen data. For both the training and testing, there are notable differences when examining the confusion matrices for the binary classifiers (Table 3). The binary classifier for performance and effort performed differently for student cohorts that were admitted on 2011 and on 2018 performed in terms of accuracy, precision and recall while the results were similar for the academic dimension. The model performed better on the 2011 dataset while in terms of recall the model performed better on the 2018 dataset. Recall is important here as the objective is to determine the students who are likely to dropout (positive class) and reduce the false negative outcomes (that is, students who were predicted as not at risk of dropping out but actually dropped out). Higher precision in 2011 test set indicates the models' dropout prediction inability as the model seem to retain less relevance with older data, (like, academic year 2011), resulting in lower false positives and increasing the overall precision value. On the contrary, higher recall in 2018 dataset indicates the model's better fit with recent data, thus contributing in lower false negatives. However, we acknowledge that 2018 is fairly recent and some students enrolled in that year might not have dropped out yet, leading to inaccurate results. As for accuracy, there are 36.05% (3273 out of 9078) instances are dropout (positive label) in the training dataset resulting in an imbalanced label distribution. The proof of concept analysis supports the hypothesis of the paper that age of the data affects the models' performance, therefore models' trained on newer data is important to increase the performance. We argue that this finding may suggest that the age of the data is pivotal to training predictive models. For the academic dimension, the overall performance was poor for both student cohorts.

5. CONCLUSION

In this paper, we explored the impact of log-based, engineered features that can be extracted from recorded student data on predicting student dropouts in Higher Education. In particular, we focused on investigating potential interactions between the engineered features and time – as represented by students' admission year – that may affect the performance of student dropout predictive models. We argued that the age of the data we use to train machine-learning models for predicting dropouts will impact the models' performance since socio-economic and cultural conditions, that arguably affect student retention, can change over time. For the purpose of this research, we engineered three sets of student features from data collected in the SIS of the HEI: one set describing the academic background of students, one set describing the performance of students and one describing the effort students put in their studies. To explore relationship between dropouts and features, and relationship between features and admission year we combined correlation analysis, MANOVA and regression analysis with admission year as the interaction term.

The results suggested that the admission year can play a critical role on the importance of the selected features for predicting dropouts. The importance of the features may change based on the socio-economic status of the state [11] which is subject to changes for multiple reasons, such as political functions, joining an economic trade or alliance, or even cultural changes and emergency situations, such as the COVID pandemic. For example, in our case, this is demonstrated by the importance of financial support provided by the state on dropout rates over the years that seems to be decreasing. One can argue that student dropouts in Higher Education is a complex topic that extends beyond the academic institution and the students themselves but it reflects socio-economic, cultural and political aspects of the society or the state. Thus, we would expect that the predictive power of engineered student features relating to societal or financial aspects – such as, the academic decisions students' make in terms of investing effort and financial support – are susceptible to change over time. On the other hand, performance-related features (such as grades and positive or negative exam results) do not appear to interact with admission year but instead their effect remains steady over student cohorts, confirming prior work [14]. Features that aim to represent the students' academic background may relate to some extent to student dropouts – as the regression analysis suggested – but their predictive power is limited and their dependency on admission year requires further investigation. To demonstrate the impact of time of predictive performance, we presented an example where we trained a binary classifier using time-sensitive features and we tested its performance on unseen data from two student cohorts that were admitted in the same HEI with a 7-year difference.

As a practical implication of this work, we envision establishing time-sensitive, predictive models for addressing student dropouts. Towards that direction, one approach would be to limit the datasets used for model training with respect to the chronology of the data, resulting in fewer older data as new data are received. However, this could lead to insufficient amount of data for training purposes. Another approach would be to incorporate "forgetting" factors in order to minimize the impact of old, non-relevant data. In this case, forgetting could be implemented by applying weights to the training set in such a way so that temporally distant or temporally irrelevant data receive lower weights (and thus, have less impact on the training) than recent entries. Similarly, for random forest or decision trees models one could regulate the threshold limits for early stopping in tree growth as a means to include the forgetting factor.

In this research, we carried out our analysis on data collected during the past decade from the same institution. This does not allow us to generalize our findings across various tem-

	MANOVA		Regression Analysis			Regression with Interaction Term		
Feature Name	F value	$Pr(>F)$	coef	std err	$Pr(>\|z\|)$	coef	std err	$Pr(>\|z\|)$
Academic Background								
normalized_score	11.88	**5.9e-4**	-0.55	0.22	**0.01**	-3.5e+2	1.94e+2	0.07
admission.special.conditions	8.02	**4.7e-3**	0.03	0.29	0.26	1.7e+2	2.7e+2	0.52
prev.study.level_Masters	4.58	**0.03**	-0.57	0.28	**0.05**	2.7e-1	1.5e-1	0.072
nr.of.prev..studies.in.UT	8.40	**3.8e-3**	0.10	0.06	**0.079**	2.3e+2	6.7e+1	**7.4e-4**
Performance								
negative.results	56.787	**6.5e-14**	0.19	0.14	0.18	-7.1e-2	2.8e-1	0.80
grade.A	52.37	**5.9e-13**	-0.12	0.06	**0.05**	1.99e-3	4.3e-2	0.96
grade.B	225.32	**<2.2e-16**	-0.02	0.06	0.73	-5.5e-3	5.0e-2	0.91
grade.C	226.77	**< 2.2e-16**	-0.06	0.06	0.30	-1.6e-2	4.9e-2	0.75
grade.D	125.77	**< 2.2e-16**	-0.10	0.07	0.16	-2.2e-2	6.3e-2	0.73
grade.E	32.16	**1.6e-08**	-0.09	0.08	0.23	-7.3e-2	8.5e-2	0.39
grade.F	3.55	0.06	0.14	0.08	0.08	1.6e-3	5.96e-2	0.99
credits.earned	1685	**< 2.2e-16**	-0.01	0.01	**0.02**	1.6e-3	5.1e-3	0.75
extracurricular.credits.earned	76.613	**< 2.2e-16**	-2.6e-3	0.01	0.75	6.2e-3	7.5e-3	0.41
not.present	173.25	**< 2.2e-16**	-1.7e-3	0.01	0.87	-7.3e-3	8.1e-3	0.36
sum_passed_grade	1381.2	**< 2.2e-16**	0.18	0.14	0.19	-4.2e-2	2.9e-1	0.88
sum_failed_grade	108.05	**< 2.2e-16**	-0.01	0.02	0.66			
all.results	1455.1	**< 2.2e-16**	-0.13	0.13	0.31	4.94e-2	2.8e-1	0.86
Effort								
days.on.academic.leave	173.41	**<2.2e-16**	2.23	8.38e-2	**<2e-16**	7.5e-4	1.6e-4	**2.6e-6**
on.extended.study.period	253.53	**<2.2e-16**	2.16e-01	4.60e-02	**2.6e-6**	1.8e-2	4.1e-2	0.66
days.studying.abroad	240.55	**<2.2e-16**	-1.34e-02	1.37e-03	**<2e-16**	-1.2e-3	8.7e-4	0.17
days.as.visiting.student	6.8963	**8.7e-3**	-1.92e-3	1.6e-3	0.22			
credits.cancelled.during.2w	476.68	**<2.2e-16**	1.01e-02	1.11e-03	**<2e-16**	5.5e-4	7.1e-4	0.44
workload	6.9	**8.7e-3**	-1.1	8.5e-1	0.21			
nr.of.courses.registered	2501.5	**<2.2e-16**	-5.89e-01	2.81e-02	**<2e-16**	-1.8e-1	1.5e-2	**<2e-16**
credits.registered	2789.8	**<2.2e-16**	-3.36e-02	1.81e-03	**<2e-16**	1.1e-3	1.1e-3	0.32
nr.of.courses.with.any.grade	2774.9	**<2.2e-16**	5.44e-01	2.65e-02	**<2e-16**	1.7e-1	1.4e-2	**<2e-16**
nr.of.employment.contracts	20.657	**5.6e-6**	-6.1e-2	4.3e-2	0.16			
total_economic_support	17.14	**3.5e-5**	-3.02e-04	4.07e-05	**1.15e-13**	1.4e-4	2.4e-5	**1.8e-8**
study_period_in_years	3941.1	**<2.2e-16**	1.29	7.71e-2	**<2e-16**	-1.6e-2	4.6e-2	0.73

Table 2: The results of MANOVA with the engineered features as the dependent variables and the admission year as the independent variable, Regression Analysis without any interaction terms and with admission year as an interaction term. The features that interact with admission year on the level $p<0.05$ are presented in bold letters

Admission Year	Accuracy	Precision	Recall	F1
Academic Background				
2011	0.718	0.500	0.025	0.048
2018	0.709	0.400	0.027	0.050
Performance				
2011	0.912	0.937	0.738	0.825
2018	0.785	0.573	1.000	0.728
Effort				
2011	0.905	0.982	0.675	0.800
2018	0.889	0.725	0.987	0.836

Table 3: The perfromance metrics of the three models that predict student dropouts per dimension for two student cohorts: the cohort admitted on year 2011 and the cohort admitted on year 2018.

poral and spatial contexts. Additionally, in this work we only used basic information about students' background and study progress - excluding demographics. Further analysis on an extended dataset may reveal significant patterns on dropouts regarding cultural background or gender. However, it is important to ensure the safe and ethical use of sensitive and personal information of students and to establish that future use of the outcomes aims to support students and academic stakeholders in a fair and accountable context. In future work, we aim to design a predictive model for addressing dropouts in HE that will implement the forgetting factor based on data's recency. For triangulation, we will compare the forgetting factor's impact both for a regression model and for a random forest model and we will explore further the impact of the forgetting factor in terms of predictive accuracy, effectiveness and efficiency. Moreover, we will consider the possibility of analyzing gender segregated data to explore if the findings show gender bias [6].

6. ACKNOWLEDGEMENTS

This research was funded by the Estonian Research Council, grant number PSG286.

7. REFERENCES

[1] D. Azcona, I.-H. Hsiao, and A. F. Smeaton. Detecting students-at-risk in computer programming classes with learning analytics from students' digital footprints. *User Modeling and User-Adapted Interaction*, 29(4):759–788, 2019.

[2] D. Baneres, M. E. Rodríguez-Gonzalez, and M. Serra. An early feedback prediction system for learners at-risk within a first-year higher education course. *IEEE Transactions on Learning Technologies*, 12(2):249–263, 2019.

[3] J. P. Bean. Dropouts and turnover: The synthesis and test of a casual model of student attrition. *Research in Higher Education*, 12(2):155–187, June 1980.

[4] I. Chounta, M. Pedaste, and K. Saks. Behind the scenes: Designing a learning analytics platform for higher education. In *Companion Proceedings of the 9th International Conference on Learning Analytics and Knowledge, Tempe, USA*, 2019.

[5] I.-A. Chounta, K. Uiboleht, K. Roosimäe, M. Pedaste, and A. Valk. Accuracy of a cross-program model for dropout prediction in higher education. In *Companion Proceedings of the 10th International Conference on Learning Analytics & Knowledge LAK20*, pages 750–755, 2020.

[6] C. Criado-Perez. *Invisible women : data bias in a world designed for men*. Abrams Press, New York, 2019.

[7] C. Glenda, M. Heagney, and L. Thomas. Improving student retention in higher education: Improving teaching and learning. *Australian Universities' Review*, 51(1):9–18, 2009.

[8] L. Hinton. Causes of attrition in first year students in science foundation courses and recommendations for intervention. *Studies in Learning, Evaluation, Innovation and Development*, 4(2):13–26, 2007.

[9] I. Johnson. Enrollment, persistence and graduation of in-site students at a public research university: Does high school matter? *Research in Higher Education*, 49:776–793, 2008.

[10] K. Kori, M. Pedaste, H. Altin, E. Tõnisson, and T. Palts. Factors that influence students' motivation to start and to continue studying information technology in estonia. *IEEE Transactions on Education*, 59(4):255–262, 2016.

[11] I. W. Li and D. R. Carroll. Factors influencing dropout and academic performance: an australian higher education equity perspective. *Journal of Higher Education Policy and Management*, 42(1):14–30, 2019.

[12] I. W. Li and D. R. Carroll. Factors influencing dropout and academic performance: an australian higher education equity perspective. *HIGHER EDUCATION POLICY AND MANAGEMENT*, 42(1):14–30, July 2020.

[13] X. Ochoa and A. Merceron. Quantitative and qualitative analysis of the learning analytics and knowledge conference 2018. *Journal of Learning Analytics*, 5(3):154–166, 2018.

[14] C. F. Rodriguez-Hernandez, E. Cascallar, and E. Kyndt. Socio-economic status and academic performance in higher education: A systematic review. *Educational Research Review*, 29:100305, 2020.

[15] V. Tinto. Through the eyes of students. *Journal of College Student Retention: Research, Theory & Practice*, 19(3):254–269, 2017.

[16] L. Willcoxson, J. Cotter, and S. Joy. Beyond the first-year experience: the impact on attrition of student experiences throughout undergraduate degree studies in six diverse institutions. *Studies in Higher Education*, 36(3):331–352, February 2011.

APPENDIX

Feature	Description
Academic Background	
normalized_score	The admission score normalized by the min and max score
admission.special.conditions	Student's admission subject to special conditions
prev.study.level	Student's latest academic degree, such as high school graduate
nr.of.prev..studies.in.UT	Number of previous enrollments in the same HEI
Performance	
nr.of.courses.with.any.grade	Registered courses with any outcome (positive or negative)
credits.earned	Sum of credits the student earned
extracurricular.credits.earned	Credits for courses extra to student's curricula
all.results	Number of all results cumulatively up to today
negative.results	Number of negative results up to today
pos.results	Number of positive results up to today
grade{A, B, C, D, E, F}	Number of all grades {A, B, C, D, E, F} up to today
passed	Number of passed, non-differentiated courses up to today
not.passed	Number of not passed, non-differentiated courses up to today
not.present	Number of non-taken exams due to absence up to today
Effort	
days.on.academic.leave	Days the student was on academic leave
on.extended.study.period	1 when student was on extended study period, 0 otherwise
days.studying.abroad	Days student was studying abroad (e.g. on an Erasmus exchange)
days.as.visiting.student	Number of days as visiting student to other Estonian universities
credits.cancelled	Number of credit points that the student cancelled
nr.of.courses.registered	Number of courses the student registered
credits.registered	Number of credits the student registered
credits.fulfilled	Ratio of credits earned vs. credits registered
nr.of.employment.contracts	Number of contracts the student has with the HEI
total_financing	Total amount of stipends and allowances
study.workload	Full time or part time student
study_period_in_years	Number of years a student has been studying

Table 4: The engineered features for each dimension. By "*up to today*", we mean the date of the data collection (19 Oct. 2020)

Deep-IRT with independent student and item networks

Emiko Tsutsumi
University of
Electro-Communications
tsutsumi@ai.lab.uec.ac.jp

Ryo Kinoshita
University of
Electro-Communications
kinoshita@ai.lab.uec.ac.jp

Maomi Ueno
University of
Electro-Communications
ueno@ai.lab.uec.ac.jp

ABSTRACT

Knowledge tracing (KT), the task of tracking the knowledge state of each student over time, has been assessed actively by artificial intelligence researchers. Recent reports have described that Deep-IRT, which combines Item Response Theory (IRT) with a deep learning model, provides superior performance. It can express the abilities of each student and the difficulty of each item such as IRT. However, its interpretability and applicability remain limited compared to those of IRT because the ability parameter depends on each item. Namely, the ability estimate for the same student and time might differ if the student attempts a different item. To overcome those difficulties, this study proposes a novel Deep-IRT model that models a student response to an item by two independent networks: a student network and an item network. Results of experiments demonstrate that the proposed method improves prediction accuracy and the interpretability of earlier KT methods

Keywords

Deep Learning, Item Response Theory, Knowledge Tracing

1. INTRODUCTION

Recently, along with the advancement of online education, Knowledge Tracing (KT) has attracted broad attention for helping students to learn effectively by presenting optimal problems and a teacher's support [5, 14, 16, 22, 23, 24, 37, 39, 43, 45, 46]. Important tasks of KT are tracing the student's evolving knowledge state and discovering concepts that the student has not mastered based on the student's prior learning history data. Furthermore, predicting a student's performance (correct or incorrect responses to an unknown item) accurately is important for adaptive learning. Many researchers have developed various methods to solve KT tasks. Methods for KT are divisible into probabilistic approaches and deep-learning approaches.

For example, Bayesian Knowledge Tracing (BKT), a traditional and well known probabilistic model for KT [1, 5, 8, 14, 16, 22, 23, 26, 45], employs a Hidden Markov Model to trace a process of student ability growth. It predicts the probability of a student responding to an item correctly. Item Response Theory (IRT) [3, 34, 35], which is used in the test theory area [10, 11, 12, 13, 28, 33, 36], has come to be used for KT [6, 40]. Actually, IRT predicts a student's correct answer probability to an item based on the student's latent ability parameter and item characteristic parameters.

Actually, a learning task is associated with multiple skills. Students must master the knowledge of multiple skills to solve a task. However, BKT and IRT have a restriction by which they express only uni-dimensional ability.

To overcome the limitations, Deep Knowledge Tracing (DKT) [24] was proposed as the first deep-learning-based method. DKT employs Long short - term memory (LSTM) [27] to predict a student's performance. LSTM relaxes the restrictions of skill separation and binary state assumptions. However, the hidden states include a summary of the past sequence of learning history data in LSTM. Therefore, DKT does not explicitly treat the student's ability of each skill.

To improve the DKT performance, various deep-learning-based methods have been proposed [2, 4, 17, 19, 29, 30, 31, 38, 42, 44]. Especially, the dynamic key-value memory network (DKVMN) was developed to exploit the relations among underlying skills and to trace the respective knowledge states [46]. To trace student ability, DKVMN uses a Memory-Augmented Neural Network and attention mechanisms. Furthermore, to improve the explanatory capabilities of the parameters, Deep-IRT was proposed by combining DKVMN with an IRT module [43]. In fact, Deep-IRT can estimate a student's ability and an item's difficulty just as standard IRT models can. However, the ability parameter of the Deep-IRT depends on each item characteristic because it implicitly assumes that items with the same skills are equivalent. The assumption does not hold when the item difficulties for the same skills differ greatly. Items for the same skills which are not equivalent hinder interpretation of a student's ability estimate.

Most recently, Gosh et al. (2020) proposed attentive knowledge tracing (AKT) [7], which incorporates a forgetting function of past data to attention mechanisms. Additionally, they indicated a problem by which earlier KT methods assumed that items with the same skills are equivalent. To re-

Emiko Tsutsumi, Ryo Kinoshita and Maomi Ueno "Deep-IRT with independent student and item networks". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 510-517. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

solve that difficulty, they employed both items and skills as inputs. The predictive accuracy of a student's performance was improved by AKT. However the interpretability of the parameters is limited because it cannot express a student's ability transition of each skill.

Earlier studies have tackled to develop deep-learning-based methods to give parameter interpretability similarly to IRT models, but those studies have not achieved it for student ability parameters, which are most important for student modeling. The problem is the difficulty of incorporating the ability parameters and item parameters independently into deep-learning-based methods so as not to degrade prediction accuracy. This study addresses that problem.

Recent studies of deep learning have shown that redundancy of parameters for training data reduces generalization error, contrary to Occam's razor. The studies also clarify the reasons [9, 20, 21]. Based on state-of-the-art reports, this study proposes a novel Deep-IRT that models a student's response to an item by two independent redundant networks: a student network and an item network. The proposed method learns student parameters and item parameters independently to avoid impairing the predictive accuracy. A student network employs memory network architecture to reflect dynamic changes of student abilities as DKVMN does. Therefore, the ability parameters of the proposed method do not depend on each item characteristic. They have higher interpretability than those of Deep-IRT. Moreover, the proposed method employs both items and skills as inputs in a different mode of Gosh et al. (2020) [7]. Although Tsutsumi et al. previously proposed a Deep-IRT for test theory, it cannot be applied to KT because a student's ability is constant throughout a learning process [32].

2. RELATED WORK

2.1 Item response theory

There are many item response theory (IRT) models [3, 18, 34, 35, 41]. This subsection briefly introduces two-parameter logistic model (2PLM): an extremely popular IRT model. In 2PLM, the probability of a correct answer given to item j by student i with ability parameter $\theta_i \in (-\infty, \infty)$ is assumed as

$$P_j(\theta_i) = \frac{1}{1 + exp(-1.7a_j(\theta_i - b_j))}, \tag{1}$$

where $a_j \in (0, \infty)$ is the j-th item's discrimination parameter expressing the discriminatory power for student's abilities, and $b_j \in (-\infty, \infty)$ is the j-th item's difficulty parameter representing the degree of difficulty.

2.2 Dynamic key-value memory network

The salient feature of DKVMN is that it assumes N underlying skills and relations between the input (items). Underlying skills are stored in key memory $\boldsymbol{M}^k \in \mathbb{R}^{N \times d_k}$. However, value memory $\boldsymbol{M}_t^v \in \mathbb{R}^{N \times d_t}$ holds abilities of underlying skills at time t. Here, d_k and d_t are tuning parameters. To express the j-th item, the input of DKVMN is a one-hot vector $\boldsymbol{q}_j \in \{0, 1\}^J$, where J represents the number of items for which the j-th element is 1 and for which the other elements are zeroes. DKVMN predicts the performance of item j at time t as explained below.

First, DKVMN calculates the attention, which indicates how strongly an item j is related to each skill as

$$\boldsymbol{\beta}_1^{(j)} = \boldsymbol{W}^{(\beta_1)}\boldsymbol{q}_j + \boldsymbol{\tau}^{(\beta_1)}, \tag{2}$$

$$w_{tl} = Softmax\left(\boldsymbol{M}_l^k \boldsymbol{\beta}_1^{(j)}\right), \tag{3}$$

where M_l^k represents a l th row vector and w_{tl} signifies the degree of strength of the relation between skill l and item j addressed by a student at time t. In addition, $\boldsymbol{W}^{(\cdot)}$ is the weight matrix and weight vector. $\boldsymbol{\tau}^{(\cdot)}$ is the bias vector and scalar. Next, student vector $\boldsymbol{\theta}_1^{(t)}$ is calculated using the weighted sum of value memory.

$$\boldsymbol{\theta}_1^{(t)} = \sum_{l=1}^{N} w_{tl} \left(\boldsymbol{M}_{tl}^v\right)^\top. \tag{4}$$

Finally, it concatenates $\boldsymbol{\theta}_1^{(t)}$ with $\boldsymbol{\beta}_1^{(j)}$ and predicts correct probability P_{tj} for an item j as

$$\boldsymbol{\theta}_2^{(t)} = \tanh\left(\boldsymbol{W}^{(\theta_2)}\left[\boldsymbol{\theta}_1^{(t)}, \boldsymbol{\beta}_1^{(j)}\right] + \boldsymbol{\tau}^{(\theta_2)}\right), \tag{5}$$

$$P_{tj} = \sigma\left(\boldsymbol{W}^{(u)}\theta_2^{(t)} + \boldsymbol{\tau}^{(u)}\right), \tag{6}$$

where $\boldsymbol{M}_{tl}^v$ represents the l th row vector of $\boldsymbol{M}_t^v$, $[\cdot]$ is a concatenation of vectors, and $\sigma(\cdot)$ represents the sigmoid function. Reportedly, DKVMN has the capability of accurately predicting performance. However, unfortunately, a lack of the interpretability of the parameter remains.

2.3 Deep-IRT

Deep-IRT is implemented by combining DKVMN with an IRT module [43] to improve the DKVMN interpretability. Deep-IRT exploits both the strong prediction ability of DKVMN and the interpretable parameters of IRT. Deep-IRT adds a hidden layer to DKVMN to gain the applicable ability and item difficulty. Specifically, when a student attempts item j at time t, an ability $\theta_3^{(t,j)}$ and item difficulty $\beta_2^{(j)}$ are calculated as shown below.

$$\theta_3^{(t,j)} = \tanh\left(\boldsymbol{W}^{(\theta_3)}\boldsymbol{\theta}_2^{(t)} + \tau^{(\theta_3)}\right), \tag{7}$$

$$\beta_2^{(j)} = \tanh\left(\boldsymbol{W}^{(\beta_2)}\boldsymbol{\beta}_1^{(j)} + \tau^{(\beta_2)}\right), \tag{8}$$

The prediction is based on the difference between $\theta_3^{(t,j)}$ and $\beta_2^{(j)}$ such as IRT.

$$P_{tj} = \sigma\left(3.0 * \theta_3^{(t,j)} - \beta_2^{(j)}\right). \tag{9}$$

Here, ability $\theta_3^{(t,j)}$ is calculated using $\boldsymbol{w}_t$ in equation (6), which depends on the item to solve because it implicitly assumes that items with the same skills are equivalent. In other words, the ability estimate for the same student and time might differ if the student attempts a different item. Furthermore, in equation (7), Deep-IRT uses item vector $\beta_1^{(j)}$ to calculate $\theta_2^{(t)}$. An important difficulty is that a student's ability, which depends on each item, hinders the interpretability of the parameters. Although Tsutsumi et al. [32] also proposed a Deep-IRT as a test theory, the purpose is different from this study because it can not be available for KT as mentioned before.

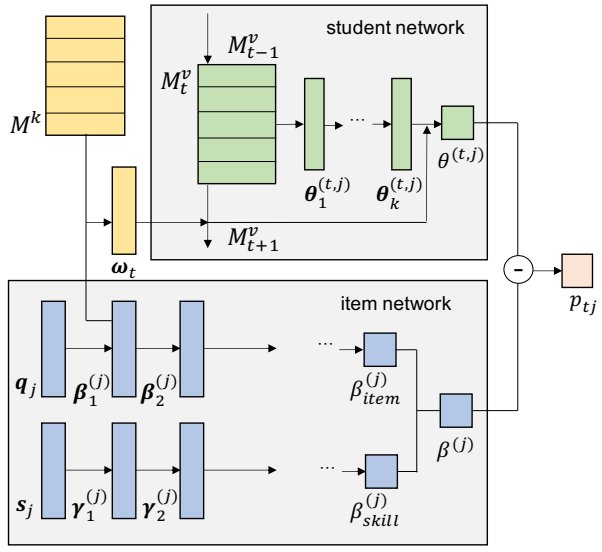


Figure 1: Network architecture for Deep-IRT with independent student and item networks. The yellow components represent the process of getting the attention weight. Also, the green components are associated with the student network and the process of updating the value memory. The blue components are associated with the item network.

3. DEEP-IRT WITH INDEPENDENT STUDENT AND ITEM NETWORKS

To resolve the difficulty described above, this study proposes a novel Deep-IRT method comprising two independent neural networks: the student network and Item deep network, as shown in Figure 1. The student network employs memory network architecture such as DKVMN to ascertain changes in student ability comprehensively. The item network includes inputs of two kinds: the item attempted by a student and the necessary skills to solve the item. Using outputs of both networks, the probability of a student answering an item correctly can be calculated.

The proposed method can estimate student parameters and item parameters independently such that prediction accuracy does not decline because the two independent networks are designed to be more redundant than with earlier methods , based on state-of-the-art reports [9, 20, 21]. The proposed method predicts P_{tj}, the probability of a correct answer assigned to item j at time t, using the item difficulties and the student abilities, as follows.

3.1 Item network

In the item network, two difficulty parameters of item j are estimated: the item characteristic difficulty parameter β^j_{item} and the skill difficulty β^j_{skill} to solve item j. The item characteristic difficulty parameter indicates the unique difficulties of the item, excepting the required skill difficulty. The proposed method expresses item difficulty as the sum of the two difficulty parameters of β^j_{item} and β^j_{skill}.

As with DKVMN, to express the j-th item, an input of the item network is a one-hot vector $\boldsymbol{q}_j \in \mathbb{R}^J$ as shown below.

$$q_{jm} = \begin{cases} 1 & (j = m) \\ 0 & (otherwise) \end{cases} \tag{10}$$

Here, J stands for the number of items. The item network comprises n layers. The item characteristic difficulty parameter of item j is calculated using a feed forward neural network as

$$\boldsymbol{\beta}^j_1 = \tanh\left(\boldsymbol{W}^{(q_1)}\boldsymbol{q}_j + \boldsymbol{\tau}^{(q_1)}\right), \tag{11}$$

$$\boldsymbol{\beta}^j_k = \tanh\left(\boldsymbol{W}^{(\beta_k)}\boldsymbol{\beta}^j_{k-1} + \boldsymbol{\tau}^{(\beta_k)}\right), \tag{12}$$

$$\beta^j_{item} = \boldsymbol{W}^{(\beta_{item})}\boldsymbol{\beta}^j_n + \tau^{(\beta_{item})}, \tag{13}$$

where $k = \{2, ..., n\}$. The last layer $\beta^{(j)}_{item}$ represents the j-th item characteristic difficulty parameter.

Similarly, to compute the difficulty of skills, the proposed method uses the input of necessary skills $\boldsymbol{s}_j \in \mathbb{R}^S$ as presented below.

$$s_{jm} = \begin{cases} 1 & (item\ j\ requires\ skill\ m) \\ 0 & (otherwise) \end{cases} \tag{14}$$

Here, S represents the number of skills:

$$\boldsymbol{\gamma}^j_1 = \tanh\left(W^{(\gamma_1)}\boldsymbol{s}_j + \tau^{(\gamma_1)}\right), \tag{15}$$

$$\boldsymbol{\gamma}^j_k = \tanh\left(W^{(\gamma_k)}\boldsymbol{\gamma}^j_{k-1} + \tau^{(\gamma_k)}\right), \tag{16}$$

$$\beta^j_{skill} = \boldsymbol{W}^{(\beta_{skill})}\boldsymbol{\gamma}^j_n + \tau^{(\beta_{skill})}, \tag{17}$$

where $k = \{2, ..., n\}$. The last layer β^j_{skill} denotes the difficulty parameter of the required skills to solve the j-th item.

3.2 Student network

In the student network, the proposed method calculates $\boldsymbol{\theta}^t_1$ based on the past response history as

$$\boldsymbol{\theta}^{(t,j)}_1 = \sum_{l=1}^{N} \boldsymbol{M}^v_{t,l}, \tag{18}$$

where $\boldsymbol{M}^v_t$ is a memory matrix holding a students' latent knowledge state, which are estimated similarly to DKVMN. Next, an interpretable student's ability vector θ^t_n is estimated as follows. Therein, n represents a number of hidden layers decided depending on the prediction accuracy of actual data.

$$\boldsymbol{\theta}^{(t,j)}_k = \tanh\left(W^{(\theta_k)}\boldsymbol{\theta}^{(t,j)}_{k-1} + \tau^{(\theta_k)}\right), \tag{19}$$

$$\theta^{(t,j)} = \boldsymbol{w}^\top_t \boldsymbol{\theta}^{(t,j)}_k, \tag{20}$$

where $k = \{2, ..., n\}$. As a difference between the proposed method and Deep-IRT, the proposed method does not multiply the attention in equation (18). In addition, $\boldsymbol{\theta}^{(t,j)}_k$ is not calculated using features of items such as equations (5) and (7). Therefore, the ability parameter vector $\theta^{(t,j)}$ does not depend on each item. Namely, it is independent from the difficulty parameter. The value of which denotes the ability for the corresponding latent skill because it is independent of any item. Therefore, $\boldsymbol{\theta}^{(t,j)}_k$ can be interpreted as a measurement model such as a multidimensional IRT [25].

3.3 Prediction of student response to an item

The proposed method predicts a student's response probability to an item using the difference between a student's ability $\theta^{(t,j)}$ to solve item j at time t and the sum of two difficulty parameters β^j_{item} and β^j_{skill}.

$$P_{tj} = \sigma\left(3.0 * \theta^{(t,j)} - (\beta^j_{item} + \beta^j_{skill})\right). \quad (21)$$

After the procedure, the value memory is updated using $\boldsymbol{c}_j$ based on the input $\boldsymbol{q}_j$ and actual performance such as DKVMN [46].

The loss function of the proposed method employs cross-entropy, which reflects classification errors. The cross-entropy of the predicted responses P_{tj} and the true responses u_{tj} is calculated as

$$\ell(u_t, P_{tj}) = -\sum_t \left(u_{tj} \log P_{tj} + (1 - u_{tj}) \log(1 - P_{tj})\right), \quad (22)$$

where u_{tj} is the true response to item j at time t. The student's response u_{tj} is recorded as 1 when the student answers the item correctly and 0 otherwise. All parameters are learned simultaneously using a well known optimization algorithm: adaptive moment estimation [15].

4. PREDICTIVE ACCURACY

4.1 Datasets

We conduct experiments to compare the performance of our approach against existing solutions. This section presents comparison of the prediction accuracies for student performance of the proposed method with those of earlier methods (DKT, DKVMN, Deep-IRT, AKT) using four benchmark datasets as ASSISTments2009[1], ASSISTments2015[2], Statics2011 [3], KDDcup[4]. ASSISTments2009 and KDDcup have item and skill tags, although most methods explained in the relevant literature adopt only the skill tag as an input. However, methods with skill inputs rely on the assumption that items with the same skill are equivalent [7]. That assumption does not hold when an item's difficulties in the same skill differ greatly. Therefore, as inputs to AKT and the proposed method, we employ not only skills but also items.ASSISTments2015 has only the skill tag. Therefore, we employ only the skill tag as an input.

Table 1 presents the number of students (No. Students), the number of skills (No. Skills), the number of items (No. Items), the rate of correct responses (Rate Correct), the average length of items which students addressed (Learning length), and the rate of items in which the number of student addressed is less than 10 (Sparsity). For all the datasets, we excepted students who addressed fewer than five items. Additionally, we set 200 items as the upper limit of the input length according to an earlier study [43]. When the input length of items becomes greater than 200, we use the first 200 response data for all methods.

[1]https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data

[2]https://sites.google.com/site/assistmentsdata/home/2015-assistments-skill-builder-data

[3]https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507

[4]https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

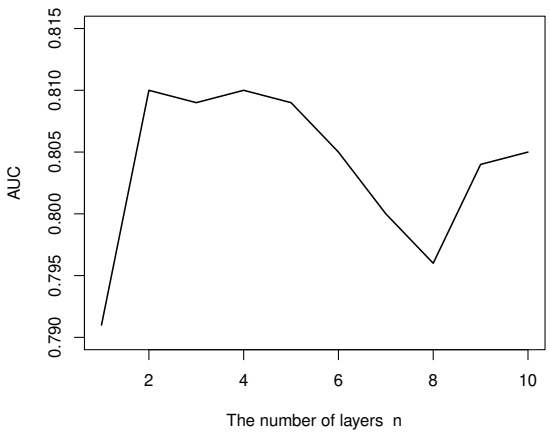


Figure 2: AUC and the Number of Layers

4.2 Hyperparameter selection and evaluation

We used ten-fold cross-validation to evaluate the prediction accuracies of the methods. The item parameters and the hyperparameters are learned using 70% of datasets. Given the estimated hyperparameters, a student's ability can be estimated at each time using the remaining 30% of each dataset. For all methods, the hidden layer size and memory dimension are chosen from $\{10, 20, 50, 100, 200\}$ using cross-validation. In addition, for the earlier methods, we used the hyperparameters reported from earlier studies [7, 43].

To ascertain the number of layers n for the proposed method, we conducted some experiments to gain experience using ASSISTments2009 while changing the layer number. The results are presented in Figure 2. As shown in the figure, AUC score reaches its highest level when $n = 2$ and $n = 4$. Based on this result, we employ $n = 2$ for the following experiments because the computation time of the proposal increases exponentially as the number of layers increases.

If the predicted correct answer probability for the next item is 0.5 or more, then the student's response to the next item is predicted as correct. Otherwise, the student's response is predicted as incorrect. For this study, we leverage three metrics for prediction accuracy: Accuracy (Acc) score, AUC score, and F1 score. The first, Acc, represents the concordance rate between the student predictive performance and the true performance. The second, AUC, represents the predictive accuracy of the correct answer probabilities. F1 indicates the average of the F1 score of incorrect answer prediction and the F1 score of correct answer prediction.

4.3 Results

The respective values of Acc, AUC, and F1 for those benchmark datasets are shown in Table 2. Results show that the proposed method with item and skill inputs provides the best performance for the metrics: averages of Acc and F1. Especially noteworthy is that the proposed method outperforms AKT, which is the most advanced method. Furthermore, the proposed method with item and skill inputs provides better performance than that with skill or item inputs. These results indicate that parameter estimation, not only with skill but also with item, improves the predictive accuracy.

Table 1: Summary of Benchmark Datasets

Dataset	No. students	No. skills	No. Items	Rate Correct	Learning Length	Sparsity
ASSIST2009	4,151	111	26,684	68.0%	70.8	55.2%
ASSIST2015	19,840	100	N/A	73.2%	34.2	12.6%
Statics2011	333	156	1,223	77.7%	180.9	2.6%
KDDcup	820	43	476	78.3%	11.9	57.8%

Table 2: Predictive Accuracy of Student Performance with Benchmark Datasets

		DKT	DKVMN	Deep-IRT	AKT	AKT (item&skill)	Proposed	Proposed (item&skill)
ASSIST2009	Acc	0.759	0.763	**0.768**	0.692	0.755	**0.768**	0.765
	AUC	0.781	0.807	0.806	0.717	0.811	**0.818**	0.810
	F1	0.697	0.714	0.718	0.639	**0.726**	0.725	0.722
ASSIST2015	Acc	0.754	0.749	0.747	**0.757**	N/A	0.752	N/A
	AUC	0.730	0.732	0.727	**0.760**	N/A	0.751	N/A
	F1	0.433	0.541	0.540	**0.616**	N/A	0.543	N/A
Statics2011	Acc	0.769	0.805	0.817	0.809	0.818	0.819	**0.822**
	AUC	0.666	0.819	0.822	0.821	**0.827**	0.821	0.821
	F1	0.483	0.679	0.681	**0.690**	0.677	0.679	**0.690**
KDDcup	Acc	0.784	0.773	0.792	0.774	0.780	0.786	**0.802**
	AUC	0.538	0.594	0.588	0.606	**0.610**	0.588	**0.610**
	F1	0.439	0.439	0.455	0.441	0.449	0.469	**0.478**
Average	Acc	0.767	0.773	0.781	0.758	0.784	0.781	**0.796**
	AUC	0.679	0.738	0.736	0.726	**0.749**	0.745	0.747
	F1	0.513	0.593	0.599	0.597	0.617	0.604	**0.630**

However, AKT with item and skill inputs shows the best average values of AUC. Actually, AKT with item and skill inputs also provides higher performance than that achieved with skill or item inputs, as shown in [7]. Gosh et al. (2020) reported that AKT is more effective for large datasets. Therefore, AKT provides the best performance for all the metrics of ASSISTments2015, which has an extremely large number of students.

Furthermore, surprisingly, the averages of ACC, AUC, and F1 obtained using the proposed method with skill input are better than Deep-IRT, although the proposed method separates student and item networks. This result implies that redundant deep student and item networks function effectively for performance prediction. These results are explainable from reports of state-of-the-art methods [9, 20, 21].

The performance results obtained using DKVMN are almost identical to those obtained using Deep-IRT because they have almost identical network structures. Results show that DKT provides the worst performance among the methods studied here.

5. PARAMETER INTERPRETABILITY

5.1 Interpretability of difficulty parameters

To evaluate the interpretability of the difficulty parameters of the proposed method, we compare the parameters of IRT with those of Deep-IRT using a simulation data. The dataset includes 2000 students' responses to 50 items and it is generated from 2PLM as shown in equation (1). The priors of the parameters have $\boldsymbol{\theta} \sim N(0,1), \boldsymbol{a} \sim LN(0,1), \boldsymbol{b} \sim N(0,1)$. We estimated the parameters of the proposal and Deep-IRT using the dataset. Table 3 shows the Pearson correlation between the true parameters of the true models and the estimated parameters, respectively, of the proposed method and Deep-IRT. Additionally, we show the prediction accuracies of the proposed method and Deep-IRT for the dataset. The proposal provides higher correlations with true parameters than Deep-IRT does, whereas the proposed method has higher accuracy than Deep-IRT has. The results demonstrate that the two independent networks of the proposed method function effectively for the interpretability of the estimated parameters and for the prediction accuracies.

Table 3: Pearson correlation

parameter	Deep-IRT	Proposed
difficulty	0.611	0.886
accuracy	0.694	0.695

5.2 Student ability transitions

This section shows student ability transitions using the proposed method. Visualizing the ability transition for each skill is helpful for both students and teachers because they can discover student strengths and weaknesses and can improve the learning method to fill in the learning gaps. Yeung [43] demonstrated a student ability transition for each skill using Deep-IRT. However, their results included some counter-intuitive ability estimates. For example, even when the student answered incorrectly, the corresponding student ability estimate increased. Moreover, Deep-IRT cannot identify a relation among multidimensional skills. There are cases in which a student's ability for low-level skills decreases even when the student responds correctly to items for high-level skills. These unstable behaviors of Deep-IRT might engender serious difficulties, which will consequently confuse students and teachers, as a student model.

Figure 3 depicts a student's ability transitions of the proposal for the ASSIST2009 dataset. The vertical axis shows

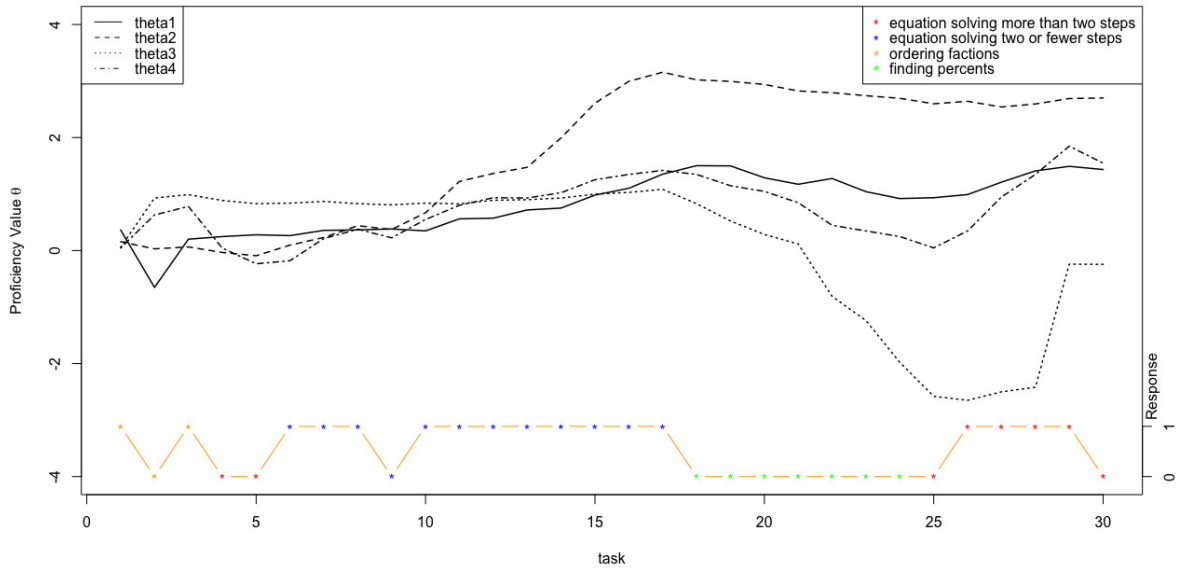


Figure 3: An example of a student ability transition from the ASSIST2009 dataset. The skill tags are classified respectively as equation solving two or fewer steps (blue), ordering fractions (orange), finding percents (green), and equation solving more than two steps (red). The student responses to items are shown at the bottom of the graph.

the student ability on the left side, with the student's response to an item on the right side. The horizontal axis shows the item number. The student's response is 1 when the student answers the item correctly; it is 0 otherwise. The student attempted skills of "equation solving more than two steps" (shown in red), "equation solving two or few steps" (shown in blue), "ordering factions" (shown in orange), and "finding percents" (shown in green). Figure 3 can be interpreted as explained below.

1. Theta 1 decreases when the student responds to item 2 "ordering factions" (orange) incorrectly and it increases when the student responds to item 3 correctly. Therefore, theta 1 indicates the ability of "ordering factions".
2. Items 6–17 correspond to the skill of "equation solving two or few steps"(blue). Theta 2 indicates the ability of "equation solving two or few steps" because theta 2 greatly increases while the student answers correctly.
3. For the skill of "finding percents" (green), the student answers all items incorrectly. Theta 3 indicates the ability of "finding percents" (green) because it greatly decreases in items 18–24.
4. Items 4, 5, and 25–30 correspond to the skill of "equation solving more than two steps" (red). Theta 4 decreases when the student answers to item 4 and 5 incorrectly, and increases when the student answers to items 26–29 correctly. Therefore, theta 4 represents the ability of "equation solving more than two steps" (red).

Figure 3 shows that the proposed method estimates the ability of each skill to reflect the student responses. Additionally, it estimates relations among the skills. Therefore, when a student responds to an item correctly/incorrectly, not only does the corresponding skill ability increase/decrease; those for other skills increase/decrease as well. Consequently, the results demonstrate that the proposed method improves both the interpretability and the prediction accuracies of Deep-IRT.

6. CONCLUSIONS

This study proposed a novel Deep-IRT that models a student's response to an item by two independent redundant networks: a student network and an item network. Because two independent redundant neural networks are used, the parameters of the proposed method can be highly interpreted with keeping hight prediction accuracy. Moreover, the proposed method employs both items and skills as inputs. Experiments demonstrated that the proposed method with item and skill inputs provided the best performance for the metrics: averages of Acc and F1. deep-learning-based methods. The result also showed AKT with item and skill inputs provided the best average values of AUC. Especially, AKT provided the best performances for large datasets as Gosh et al. (2020) reported [7]. In addition, results of experiments show that the parameters of the proposed method are more interpretable than those of Deep-IRT. This study employed slightly redundant deep networks compared to earlier methods. As future work, we intend to use the proposed method to investigate the performances of more redundant and deeper networks. In addition, we will try to optimize a forgetting function for past data to maximize the prediction accuracy for large data sets.

7. ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Numbers 19H05663 and 19K21751.

8. REFERENCES

[1] D. Agarwal, R. Baker, and A. Muraleedharan. Dynamic knowledge tracing through data driven recency weights. In *Proceedings of the 13th International Conference on Educational Data Mining, EDM*, pages 725–729, 2020.

[2] F. Ai, Y. Chen, Y. Guo, Y. Zhao, Z. Wang, G. Fu, and G. Wang. Concept-aware deep knowledge tracing and exercise recommendation in an online learning system. In *EDM*, 2019.

[3] F. Baker and S. Kim. *Item Response Theory: Parameter Estimation Techniques, Second Edition*. Statistics: A Series of Textbooks and Monographs. Taylor & Francis, 2004.

[4] P. Chen, Y. Lu, V. Zheng, and Y. Pian. Prerequisite-driven deep knowledge tracing. In *In IEEE International Conference on Data Mining, ICDM 2018*, pages 39–48, 2018.

[5] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Model. User-Adapt. Interact.*, 4(4):253–278, Dec 1995.

[6] C. Ekanadham and Y. Karklin. T-skirt: Online estimation of student proficiency in an adaptive learning system. *CoRR*, abs/1702.04282, 2017.

[7] A. Ghosh, N. Heffernan, and A. S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[8] S. Gowda, J. Rowe, R. Baker, M. Chi, and K. Koedinger. Improving models of slipping, guessing, and moment-by-moment learning with estimates of skill difficulty. In *EDM 2011 – Proceedings of the Fourth International Conference on Educational Data Mining*, pages 199–208, 01 2011.

[9] H. He, G. Huang, and Y. Yuan. Asymmetric valleys: Beyond sharp and flat local minima. In *Advances in Neural Information Processing Systems 32*, pages 2553–2564. Curran Associates, Inc., 2019.

[10] T. Ishii, P. Songmuang, and M. Ueno. Maximum clique algorithm for uniform test forms assembly. In *The 16th International Conference on Artificial Intelligence in Education*, volume 7926, pages 451–462, 07 2013.

[11] T. Ishii, P. Songmuang, and M. Ueno. Maximum clique algorithm and its approximation for uniform test form assembly. *IEEE Transactions on Learning Technologies*, 7:83–95, 01 2014.

[12] T. Ishii and M. Ueno. Clique algorithm to minimize item exposure for uniform test forms assembly. In *International Conference on Artificial Intelligence in Education*, pages 638–641, 06 2015.

[13] T. Ishii and M. Ueno. Algorithm for uniform test assembly using a maximum clique problem and integer programming. In *Artificial Intelligence in Education. Springer International Publishing*, pages 102–112, 06 2017.

[14] M. Khajah, Y. Huang, J. Gonzalez-Brenes, M. Mozer, and P. Brusilovsky. Integrating knowledge tracing and item response theory: A tale of two frameworks. *Personalization Approaches in Learning Environments*, 1181:5–17, 2014.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the ICLR*, 2014.

[16] J. Lee and E. Brunskill. The impact on individualizing student models on necessary practice opportunities. In *Proceedings of the Fifth International Conference on Educational Data Mining*, pages 118–125, 01 2012.

[17] X. Liangbei and D. Mark. Dynamic knowledge embedding and tracing. In *Proceedings of the 13th International Conference on Educational Data Mining, EDM*, pages 524–530, 2020.

[18] F. Lord and M. Novick. *Statistical Theories of Mental Test Scores*. Addison-Wesley, 1968.

[19] Y. Lu, D. Wang, Q. Meng, and P. Chen. Towards interpretable deep learning models for knowledge tracing. In *Proceedings of the 13th International Conference on Educational Data Mining, EDM*, pages 185–190, 2020.

[20] A. Morcos, H. Yu, M. Paganini, and Y. Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *Advances in Neural Information Processing Systems 32*, pages 4932–4942. Curran Associates, Inc., 2019.

[21] V. Nagarajan and J. Z. Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems 32*, pages 11615–11626. Curran Associates, Inc., 2019.

[22] Z. Pardos and N. Heffernan. T.: Modeling individualization in a bayesian networks implementation of knowledge tracing. In *In Proceedings of the 18th International Conference on User Modeling, Adaption, and Personalization*, pages 255–266, 06 2010.

[23] Z. Pardos and N. Heffernan. Kt-idem: Introducing item difficulty to the knowledge tracing model. In *Proceedings of 19th International Conference on User Modeling, Adaptation and Personalization (UMAP 2011)*, pages 243–254, 01 2011.

[24] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 505–513. Curran Associates, Inc., 2015.

[25] M. Reckase. Multidimensional item response theory models, springer. 2009.

[26] J. Reye. Student modelling based on belief networks. *International Journal of Artificial Intelligence in Education*, 14:63–96, 2004.

[27] H. Sepp and S. Jurgen. Long short-term memory. *Neural Computation*, 14:1735–1780, 1997.

[28] P. Songmuang and M. Ueno. Bees algorithm for construction of multiple test forms in e-testing. *IEEE Transactions on Learning Technologies*, 4:209–221, 07 2011.

[29] Y. Su, Q. Liu, Q. Liu, Z. Huang, Y. Yin, E. Chen, C. H. Q. Ding, S. Wei, and G. Hu. Exercise-enhanced sequential modeling for student performance prediction. In *AAAI*, pages 2435–2443, 2018.

[30] X. Sun, X. Zhao, Y. Ma, X. Yuan, F. He, and J. Feng. Multi-behavior features based knowledge tracking

using decision tree improved dkvmn. In *Proceedings of the ACM Turing Celebration Conference – China*, New York, NY, USA, 2019. Association for Computing Machinery.

[31] H. Tong, Y. Zhou, and Z. Wang. Exercise hierarchical feature enhanced knowledge tracing. In *Proceedings of the 13th International Conference on Educational Data Mining, EDM*, pages 324–328, 2020.

[32] E. Tsutsumi, R. Kinoshita, and M. Ueno. Deep item response theory as a novel test theory based on deep learning. *Electronics*, 10(9), 2021.

[33] M. Ueno. Adaptive testing based on bayesian decision theory. *International Conference on Artificial Intelligence in Education*, pages 712–716, 2013.

[34] M. Ueno and Y. Miyazawa. Probability based scaffolding system with fading. *Artificial Intelligence in Education – 17th International Conference, AIED*, pages 237–246, 2015.

[35] M. Ueno and Y. Miyazawa. Irt-based adaptive hints to scaffold learning in programming. *IEEE Transactions on Learning Technologies*, 11(4):415–428, Oct 2018.

[36] M. Ueno and S. Pokpong. Computerized adaptive testing based on decision tree. In *Advanced Learning Technologies (ICALT), 2010 IEEE Tenth International Conference*, pages 191–193, 2010.

[37] X. Wang, J. Berger, and D. Burdick. Bayesian analysis of dynamic item response models in educational testing. *The Annals of Applied Statistics*, 7(1):126–153, 2013.

[38] Z. Wang, X. Feng, J. Tang, G. Huang, and Z. Liu. Deep knowledge tracing with side information. In *The 20th International Conference on Artificial Intelligence in Education (AIED)*, pages 303–308, 2019.

[39] R. Weng and D. Coad. Real-time bayesian parameter estimation for item response models. *Bayesian Analysis*, 13, 12 2016.

[40] K. H. Wilson, Y. Karklin, B. Han, and C. Ekanadham. Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation. In *9th International Conference on Educational Data Mining*, volume 1, pages 539–544, 06 2016.

[41] W.J. van der Linden. *Handbook of Item Response Theory, Volume Two: Statistical Tools*. Chapman and Hall/ CRC Statistics in the Social and Behavioral Sciences. Chapman and Hall/ CRC, 2016.

[42] X. Xiong, S. Zhao, V. Inwegen, E. G., and J. E. Beck. Going deeper with deep knowledge tracing. In *Proceedings of International Conference on Education Data Mining*, 2016.

[43] C. Yeung. Deep-irt: Make deep learning based knowledge tracing explainable using item response theory. In *Proceedings of the 12th International Conference on Educational Data Mining, EDM*, 2019.

[44] C. K. Yeung and D.-Y. Yeung. Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proceedings of the Fifth ACM Conference on Learning @ Scale*, pages 1–10, 2018.

[45] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *Artificial Intelligence in Education*, pages 171–180, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[46] J. Zhang, X. Shi, I. King, and D.-Y. Yeung. Dynamic key-value memory network for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 765–774, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

Modeling Creativity in Visual Programming: From Theory to Practice

Anastasia Kovalkov
Ben-Gurion University of the Negev
Kovalkov@post.bgu.ac.il

Benjamin Paaßen
Humboldt-University of Berlin
benjamin.paassen@hu-berlin.de

Avi Segal
Ben-Gurion University of the Negev
kobig@bgu.ac.il

Kobi Gal
Ben-Gurion University of the Negev
University of Edinburgh
kobig@bgu.ac.il

Niels Pinkwart
Humboldt-University of Berlin
niels.pinkwart@hu-berlin.de

ABSTRACT

Promoting creativity is considered an important goal of education, but creativity is notoriously hard to define and measure. In this paper, we make the journey from defining a formal creativity and applying the measure in a practical domain. The measure relies on core theoretical concepts in creativity theory, namely fluency, flexibility, and originality, We adapt the creativity measure for Scratch projects. We designed a machine learning model for predicting the creativity of Scratch projects, trained and evaluated on ratings collected from expert human raters. Our results show that the automatic creativity ratings achieved by the model aligned with the rankings of the projects of the expert raters more than the experts agreed with each other. This is a first step in providing computational models for describing creativity that can be applied to educational technologies, and to scale up the benefit of creativity education in schools.

Keywords

Creativity, Creativity Tests, Visual Programming Environments

1. INTRODUCTION

Modern education generally tries to foster creativity in student problem solving [7, 11, 17]. There is wide agreement that creative solutions must not only solve the task but should additionally be *original*, i.e. distant from usual solutions to the task, *flexible*, i.e. employ very different concepts, and *fluent*, i.e. employ many concepts [10, 23]. However, creativity is notoriously hard to quantify in practice [7]. When being confronted with two student solutions for a given learning task, different teachers may well disagree which one is more creative [14].

Anastasia Kovalkov, Benjamin Paassen, Avi Segal, Kobi Gal and Niels Pinkwart "Modeling Creativity in Visual Programming: From Theory to Practice". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 518-524. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

In this paper, we make the journey from a definition of creativity which relate to prior concepts in the literature [23], to applying the definition to the Scratch programming environment, and using the measure to automatically quantifying the creativity score of projects in Scratch. We formalize originality of a product as the distance to usual solutions, flexibility as the distance between concepts in the student's solution, and fluency as the distance to an empty solution.

We apply the formalization to automatically measure the creativity on a set of projects from the popular visual programming language Scratch [13]. Using machine learning, we build a model that predicts the creativity ratings of Scratch projects using fluency, flexibility, and originality measures of our approach. We compare these automatic creativity ratings to those of five human experts, which were collected using a comprehensive user study. We find that the automatic ratings agree with the rankings of the experts more than the experts agreed with each other. We provide several examples that highlight the benefit of the model in light of the fact that human raters may disagree on the degree of creativity of Scratch projects.

The contribution of this work is in providing an automatic framework for defining and detecting creativity, that can scale up teacher's abilities to support creative thinking in students.

2. RELATED WORK

Prior works on measuring creativity have mostly been concerned with psychological tests, such as Williams' tests on creative thinking [25] or the Torrance test of creative thinking [10, 23]. However, such tests do not account for changes in creative ability, motivation, knowledge, and social context over time [2]. Accordingly, one should wish to measure creativity often and monitor the development across changing circumstances. This could be supported by automatic creativity assessment, towards which we work in this paper.

To measure creativity at one specific point in time, we follow Torrance's work and grade creativity on three scales, namely fluency, flexibility, and originality [10, 23]. Historically, these three scales grew out of Guilford's model of the structure of

intellect [6], which includes divergent production, i.e. the skill of generating a wide variety of ideas on the same topic. In particular, fluency refers to the sheer amount of ideas generated, flexibility to the number of distinct classes of ideas, and originality to the infrequency of the ideas compared to a general sample of the population. In addition to these three scales, Torrance's tests also include scales regarding the abstractness of generated ideas, the elaboration on ideas, and the resistance to premature closure during the generation process [10]. In this work, we stick to fluency, flexibility, and originality because they permit quite a direct formalization in mathematical and computeable terms. We further cover elaboration, to some extent, in our notion of fluency.

Multiple works focused on using technologies to infer creative thinking in numerous educational disciplines, such as math and programming [21, 12]. Previous research has shown that creativity is related to positive learning gains, and using technology to generate creativity is an active field of study [24]. Hershkovitz et al. [8, 9] examined the relationship between creativity and computational thinking within a block-based multi-level game environment for children's programming. Their findings show that creativity can contribute to the acquiring of computational thinking and can also be transferred across domains, stressing the importance of fostering creativity while promoting computational thinking.

Finally, the application domain of our work is Scratch, a block-based visual programming language targeted primarily at children. The environment allows users to create interactive stories, games, and animations [13]. Scratch blocks are designed to fit together in ways that make syntactic sense which generates the program logic. Users can use a wide variety of pre-defined basic code blocks, such as *When Key Pressed*, *Move* etc. Furthermore, programmers can use additional blocks such as *Pen Down* and *Language* from existing extensions such as 'Pen' and 'Translate', as well as define custom blocks. The environment allows the use of external data through importing images, music recordings, captured voices, and user-specific graphics [18]. By using Scratch, which is designed to enable creative expression in terms of code, graphical and audio aspects, we can expand the identification of creativity beyond the programming aspect [3, 5, 12].

3. COMPUTING CREATIVITY IN SCRATCH

In this section, we describe how we measure creativity of code and visual aspects of Scratch programs.

A Scratch project consists of the project's background called *stage* and the objects that appear on it called *sprites*[1]. Figure 1 presents a sample project, a game called 'Scratch in Scratch'. As its name implies, it simulates the Scratch environment. The player has to select a character and add block instances to a stack of blocks that control the character's behavior on the stage. The figure shows the white stage and different sprites (buttons, arrows and a cartoon character), as well as the graphics output area. Blocks are code elements that control the behavior of the stage and sprites [13]. When a sprite is selected, its blocks element are shown in the Code panel. Figure 1 (center) shows the blocks that are connected to the cartoon cat, whose sprite is selected (e.g., blocks *Hide* and *Show*).

Inspired by the creativity test of Torrance [23], we measure creativity on three scales: fluency, flexibility, and originality. Generally speaking, fluency refers to the amount of ideas involved in the Scratch project, flexibility to the diversity of ideas, and originality to the distance between a project and typical projects [10]. In the following, we describe how we compute these scales for code and visual aspects, respectively. For both aspects, our strategy is to first define a distance between building components (e.g. code blocks and images) and then compute fluency, flexibility, and originality based on that distance.

Code Creativity. We represent the Scratch code as a collection of syntax trees, one representing the stage, and one for each sprite. Each syntax tree, in turn, consists of code blocks. Figure 2 shows a graph of the blocks in Scratch, where blocks are connected to the semantic sub-category they belong to (like *move* or *events*) and sub-categories are connected to categories, namely basic blocks, extension blocks, and custom blocks. Let now δ be the shortest-path distance between blocks in this graph. More specifically, we define δ as zero for equal blocks (e.g., two *Move* blocks), as 1 if the blocks are different but within the same sub-category of blocks (e.g., *Move*, *Turn*), and as 2 if the blocks are different and from different sub-categories (e.g., *Move*, *When Key Pressed*). To explain, we add one unit of distance for the transition between the sub-categories and another for the blocks being different.

For different categories, the distance between the pre-defined blocks and the extension blocks is defined as 3, and the distance between the pre-defined blocks to the custom blocks is defined 4. To explain, we add one one unit of distance for the block's difference, the second unit of distance due to the different sub-categories, and the third for the category change. Since the Scratch environment provides by default the pre-defined blocks, while custom blocks require the user to build something new, we add an additional unit of distance.

Based on the distance, we define code fluency of a Scratch project as the sum $\sum_x \delta(x, 0)$, where x are the code blocks in the project and 0 is the gray zero node in Figure 2. The distance $\delta(x, 0)$ to basic blocks (e.g. *Move*) is defined as 3, the distance $\delta(x, 0)$ to extension blocks (e.g., *Pen-Up*, *Language*) is 4, and the distance $\delta(x, 0)$ to custom blocks is 5. In other words, we assign higher fluency for the production of non-existent components or the use of custom blocks that require additional user effort. For example, in the 'Scratch in Scratch!' program the *Show* block presented in Figure 1 is a basic block from the sub-category 'Looks'. The program gets 3 points for this block, and an overall fluency score of 10523[2].

To compute code flexibility, we remove all duplicated blocks,

[1] `https://www.scratch-wiki.info/`

[2] For mathematical reasons we square the distances, thus yielding large numbers.

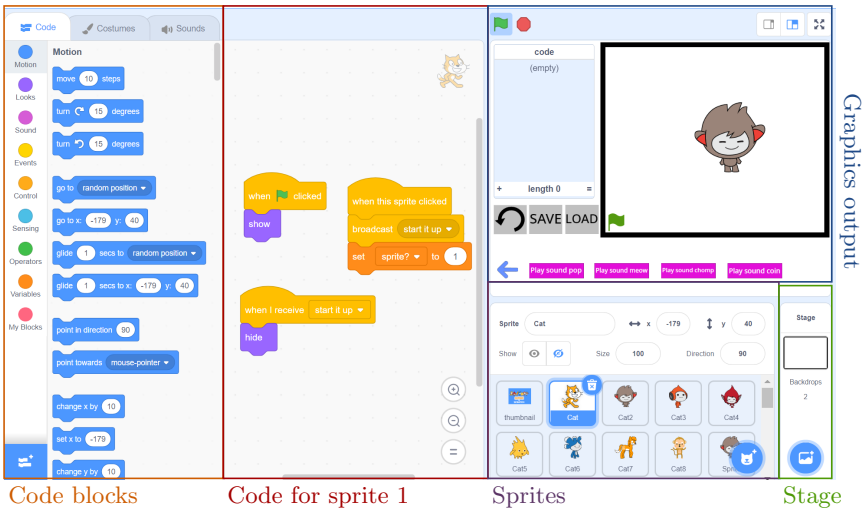


Figure 1: A screenshot of an example Scratch project. Left: A selection of possible code blocks. Center: The code blocks related to the currently selected sprite (the cat). Top right: The current graphical output. Bottom right: An overview of all sprites and the stage (i.e. the background).

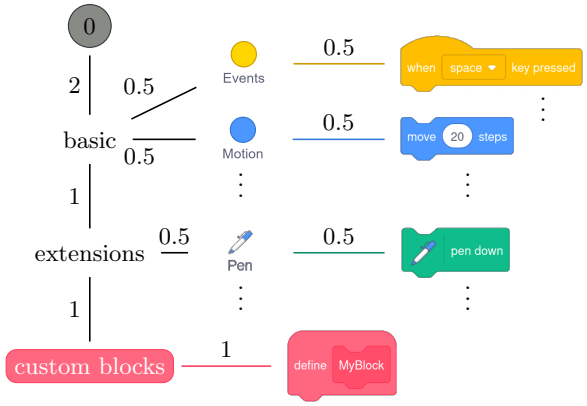


Figure 2: A graph of code blocks in scratch. We compute the distance between blocks by their shortest path distance in the graph, e.g. $\delta($  ,  $) = 0.5 + 0.5 + 1 + 0.5 + 0.5 = 3$.

then compute the sum of all pairwise distances $\sum_x \sum_y \delta(x, y)$ between code blocks in the project and divide by the number of unique code blocks. This measures how different the code blocks were, capturing the idea of flexibility as variability of concepts [10]. For example, the project 'Scratch in Scratch!' uses the blocks *Hide* and *Show* for each sprite. This increases fluency but not flexibility. Further, these two pre-defined blocks belong to the same sub-category 'Looks' and so $\delta(Hide, Show) = 1$. Overall, after normalizing by the number of the unique blocks in the program (58), it obtained a flexibility score of 395.37

We define code originality as the average distance of a Scratch project to a sample of typical projects in our data set, i.e. a project that is more distant from typical projects is considered more original. To compute the distance between projects, we follow the approach of Price et al. [22]. In particular, we use a three-step algorithm to construct an alignment between two Scratch projects. First, we compute the tree edit distance [26] between the stage syntax trees of both programs. Then, we compute all pairwise tree edit distances between sprites in both programs. Finally, we feed this result into the Hungarian algorithm [15] to obtain an alignment between the sprite trees. This is because sprites in a Scratch project do not have a clear ordering, making an unordered representation more natural.

While fluency and flexibility are based only on the program itself, the originality requires a reference sample of projects, i.e. we need a reference point with respect to which a project is original or not [23, 20]. To illustrate the effect of the reference set, we note that the originality of the 'Scratch in Scratch!' with respect to a reference set of 3 different project groups from the user study was 4488.84, 4168.89, and 2759, respectively.

Visual creativity. To represent visuals, we first collect all images (i.e. sprite and stage images) in our training data set and feed them into a ResNet50 neural network[3]. ResNet50 has been shown to generalize diverse image processing tasks and classifications [19, 16]. Accordingly, we hope that the representation of ResNet50 also helps to capture the semantic distance between images for our case. The output is a set of vectors, one for each image. To measure distance δ between images, we use the Cosine distance $\delta(x, y) = 1 - \frac{x^T \cdot y}{\|x\| \cdot \|y\|}$ because it is invariant against effects of scale/size, which would otherwise be a confounder in our data.

We compute visual fluency as the number of images in the Scratch projects, which is equivalent to the fluency definition of Torrance [23]. For example, in the project 'Scratch in

[3] `https://www.kaggle.com/keras/resnet50`

Scratch!' we have 47 images and that is its fluency score.

To measure visual flexibility, we use the same approach as for code flexibility, i.e. we compute the sum $\sum_x \sum_y \delta(x, y)$ of all pairwise distances over images in the project and then divide by the number of images. To illustrate, the program 'Scratch in Scratch' contains 2 similar button images 'Save' and 'Load' (see Figure 1). The Cosine distance between these images is relatively small $\delta('Save', 'Load') = 0.19$, based on the vectors created by the ResNet50 network. The flexibility score of the visual aspect of the project is 12.42.

We compute visual originality as the average distance of each image in a project to images in typical projects. To illustrate, the project 'Scratch in Scratch!' received an originality score of 0.57, 0.57 and 0.58 when using 3 different reference sets.

4. HUMAN CREATIVITY ASSESSMENT

In this section, we describe a user study to collect expert evaluations of creativity of Scratch projects. The experts were Scratch instructors without prior knowledge in creativity theory. Each expert was assigned a set of pre-selected Scratch projects and asked to separately evaluate the creativity of projects according to four different aspects: code, visual, audio and idea behind the project, which was identified in past work as an important factor in the creative process in Scratch [18].

We designed an online application to facilitate the rating process and to allow the experts to play and review each project as they see fit. The application was divided into three main screens. The Home Screen displays all of the projects that are assigned to an expert. When clicking on a project in the Home Screen, experts were able to see additional information about the project (e.g., the number of views and likes that the project received) and information about the user (e.g., country, date of registration in Scratch, and age if available) and also a link to the editor environment for the project's code and visuals and the embedded playable project.

For each project, experts were asked to answer questions that relate to the creativity of the four aspects of the Scratch project. Questions relating to visual aspects, such as whether the project contained images provided by scratch or originally created by the user. Experts were also asked to rate the novelty/quality and effort put into the visual aspects of the project. Questions relating to the project code, such as evaluating the code complexity, efficiency and novelty, and rating the effort put into the code. Questions about the project idea asked to include a short description of the idea and ratings for how much novelty and effort were required for developing the idea. If the project included sounds, experts were asked if these sounds were recorded by the user, imported or were provided by Scratch. Additionally, the experts rated the novelty of the sounds and the effort invested in the audio aspect.

Experts were also asked to provide a creativity score for each of the aspects (0-100), shown in Figure 3, as well as provide a weight (between 0 and 1, summing to 1) for each aspect according to its subjective importance in determining the creativity of a project. The creativity score of a given project for an expert is computed as the weighted summation of the creativity ratings for each aspect. The creativity score for each project is shown in the Home Screen, allowing experts to compare the scores and revise them at will.

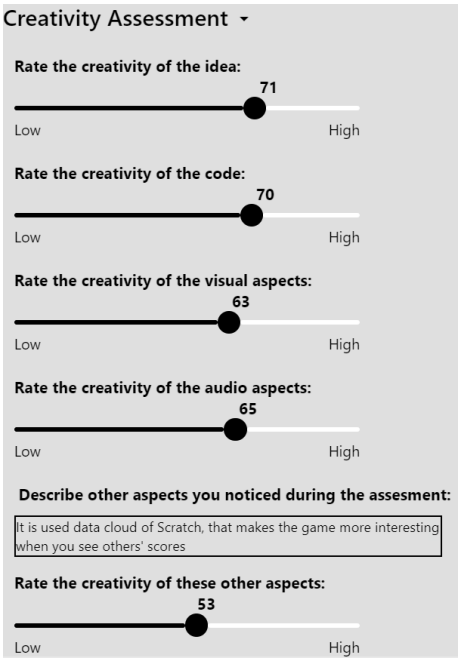


Figure 3: Overall creativity assessment.

Table 1: Experts grading statistics of code, visual and final creativity scores

Expert	Statistic	Code	Visual	Final score
1	Mean	69.55	75.85	67.59
	SD	24.04	24.97	21.31
2	Mean	66.75	67.70	66.89
	SD	13.11	14.28	13.80
3	Mean	70.75	77.65	65.30
	SD	10.18	10.50	11.89
4	Mean	72.90	83.40	76.11
	SD	24.00	20.11	17.78
5	Mean	64.60	68.55	63.52
	SD	15.27	13.15	13.17

4.1 User Study

We recruited 5 experts from 4 countries: Cuba, Vietnam, India and Israel. All experts had at least two years of Scratch training experience to students of different ages in schools and after-school activities. We selected 45 unique projects of different types (games and stories), created by different users (age ranged between 9 to 18, from 25 different countries, and with different experience, from 4 to 258 projects). We uniformly sampled projects to each of the experts from this set, so that there is a sufficient spread of creativity assessments across project, while still having some projects being rated by several experts. Four of the experts evaluated 20 projects, while one evaluated 10 projects.

Table 1 presents the statistics of the scores for code, visuals, and the final creativity scores provided by the experts. We note that the highest scores were provided by expert 4 and that this expert as well as expert 1 had the highest standard deviation across all aspects. Experts 2 and 5, on the other hand, gave relatively lower scores with a lower standard deviation.

4.2 Agreement between experts

Experts differed widely in the creativity scores they assigned to projects. For example, experts 1, 2, and 3 all evaluated the project 'Scratch in Scratch!'. Expert 1 gave this project a creativity score of 91 for the code aspect, while expert 2 gave it a score of 67, and expert 3 gave a score of 82.

We note that the low agreement between raters should not signify a mistake or lack of expertise. It reflects the fact that creativity assessment is largely subjective, and that experts can differ about which aspects are more or less important when measuring creativity, as we show in this section. To compensate for this, we measure agreement using the Kendall Rank Correlation Coefficient [1]. This measure ranges from -1 (complete disagreement between rankings) to 1 (perfect match) and is determined based on overlapping projects for each pair of experts.

Figure 4 displays for each pair of experts the number of overlapping projects (in parentheses) and the Kendall-τ score. Note that experts 2 and 4 had only one overlapping project, therefore the Kendall-τ score cannot be calculated for them. As shown in the figure, the highest agreement (Kendall-$\tau = 0.67$) is between expert 5 and 2. The other positive agreement scores are much lower and vary between 0.2 and 0.33. Moreover, we see four pairs of experts with negative Kendall-τ scores, with 2 of them including expert 4.

The experts with the highest agreement score (experts 2 and 5) also exhibited similar scores for code and visual aspects (See Table 1), suggesting that they interpret creativity for these aspects in similar ways. However, the same expert 5 commonly disagreed with expert 1 (Kendall-$\tau = -0.6$). Their scores and rankings of overlapping projects differed substantially, suggesting that they differ in their interpretation of creativity. For example for the code aspect, the same project was ranked 1st by expert 5 and 16th by expert 1.

We observe that most experts found the visual aspect more significant than the code aspect when evaluating creativity. For the majority of experts, the project idea was the most important aspect. By contrast, experts assigned low weights to audio aspects. We note that the project idea is very difficult to model computationally. This is an interesting avenue to explore in future work.

5. PREDICTING CREATIVITY SCORES

In this section we report on the design and evaluation of a computational model to predict the creativity scores of Scratch projects. We build an automatic tool to support teachers (and students) in Scratch that can be trained on examples taken from individual or multiple experts.

We use an XGBoost Regressor [4] to predict the expert creativity scores for each project. As input features we used the

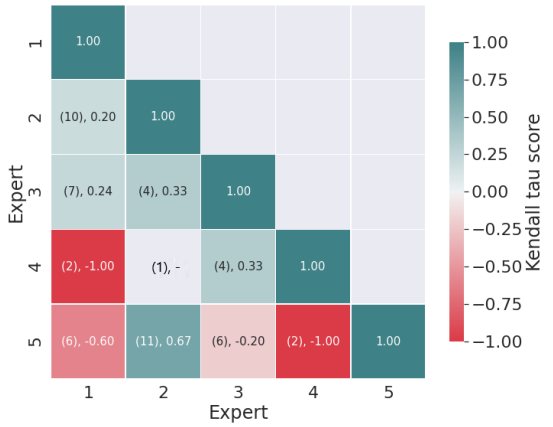


Figure 4: Kendall-τ Agreement between pairs of experts with overlapping projects on the final creativity score. (The number of overlapping projects is shown in parentheses.)

originality, flexibility, and fluency measures for both visual and code aspects, as described in Section 3. This provided us with 6 features for each instance (project). The reference sample for computing originality included all of the projects that the expert rated.

We created 2 types of XGBoost models: (1) a single rater model trained on projects for each expert separately and (2) a combined model trained on the projects from all experts together. Note that for the combined model, projects that were evaluated by more than one rater were treated as different instances. For each type of model, we created three different prediction models (a) predicting the code creativity score. (b) predicting the visual creativity score. (c) predicting the overall project score by the weighted combination score (visual and code). The features consisted of the originality, flexibility and fluency for the code aspect (model a), the visual aspect (model b), or both (model c).

The combined model and the single rater models were developed using the official implementation of XGBoost[4]. We selected the hyperparameters based on the structure of our data. We set the upper complexity limit of the model to six trees for the rater with 10 projects, 14 for the rest of the raters, and 29 trees for the combined raters and the maximum tree depth based on the number of features. The combined model was evaluated using 10-fold cross-validation. The single rater models were evaluated using 5 folds to ensure that the test set contained at least two projects.

Because of the high degree of variance between the raters, we do not seek to minimize error with respect to the predicted creativity score. Instead, we compare rankings. Ideally, we would compare the rankings of the projects in the test set with the true rankings for each expert. However, the size of the test set for some folds for some of the experts was small (4 projects for most experts). To increase the number of comparisons for Kendall-τ, we built a complete ranking

[4]`https://github.com/dmlc/xgboost`

Table 2: Kendall-τ score between XGBoost Regressor and experts scores - code creativity score, visual creativity score, and the weighed visual and code score

Experts	Kendall-τ		
Expert	Code	Visual	Weighed visual and code
1	0.52	0.52	0.42
2	0.51	0.42	0.42
3	0.53	0.58	0.36
4	0.46	0.52	0.57
5	0.52	0.42	0.50
Combined	0.43	0.44	0.42

over all projects for an expert, by combining the predicted scores of projects in the test-set with the scores of projects in the training set. However, we compute the Kendall-τ agreement only for project pairs with at least one project in the test set. We make a similar computation with respect to computing the Kendall-τ for the combined model.

Table 2 presents the Kendall-τ performance, computed as described above. As seen from the table, when predicting the creativity score for visual and code aspects, we achieve a Kendall-τ score of 0.51 and above for 3 out of 5 experts. When predicting the weighted creativity score, we achieve a Kendall-τ score of over 0.42 for 4 out of 5 experts. Overall, the agreement measure is higher than that of the inner-agreement between the experts themselves that is reported in Figure 4 (except for the pair 2 and 5).

The bottom row in Table 2 presents the results for the XGBoost model that is trained over the combined set for all experts. In all cases we achieve a Kendall-τ score above 0.42, which is higher than the inner-agreement scores for most pairs of experts. For visual aspects, the combined model is less successful than the individual models. In contrast, for the visual creativity, the combined model is better than the single rater model for two of the experts (experts 2 and 5); for weighted visual and code creativity score, the combined model is better or equal than the single rater model for 3 experts (experts 1, 2 and 3). This suggests that our models can define useful rules for aggregating creativity rankings by different experts despite the disagreement between them.

6. DISCUSSION AND CONCLUSION

In this paper we presented a formalization of creativity in terms of fluency, flexibility, and originality. We automatically computed creativity both for code and for visual aspects of Scratch projects and we intend to add the other possible modalities of that environment to our future work. Further, we set up a web application to rate the creativity in Scratch projects independently of our formalization. Finally, we recorded the ratings of five human experts on 45 Scratch projects via this application.

We observed that human raters tend to disagree on which projects are creative and which are not. Still, we were able to train regression forests, which achieved a higher ranking agreement with the human raters than they achieved with each other, and which only used the automatically generated ratings as input. We observed that the regression forest model could further improve its accuracy when being applied to individual experts instead of their shared data.

Our approach makes a step towards supporting teacher's abilities to detect and support creative outcomes in students' work. Ample future work is still to be done. Furthermore, we plan to analyze creativity ratings over time, thus tracking students' creative learning process. Future work will also need to address how an automatic assessment of creativity can support creativity at scale in technological environments, taking into account different subjective interpretations.

7. ACKNOWLEDGMENTS

This paper was made possible by funding from the German Research Foundation (DFG) under grant number PA 3460/2-1.

References

[1] H. Abdi. The Kendall rank correlation coefficient. *Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA*, 1:508–510, 2007.

[2] T. M. Amabile. *Creativity in context: Update to the social psychology of creativity.* Routledge, 2018.

[3] J. Bustillo and P. Garaizar. Using Scratch to foster creativity behind bars: Two positive experiences in jail. *Thinking Skills and Creativity*, 19:60 – 72, 2016.

[4] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proc. SIGKDD*, page 785–794, 2016.

[5] M. N. Giannakos, L. Jaccheri, and R. Proto. Teaching computer science to young children through creativity: Lessons learned from the case of Norway. In *Proc. CSERC*, page 103–111, 2013.

[6] J. P. Guilford. The structure of intellect. *Psychological bulletin*, 53(4):267–293, 1956.

[7] D. Henriksen, P. Mishra, and P. Fisser. Infusing creativity and technology in 21st century education: A systemic view for change. *Educational Technology & Society*, 19(3):27–37, 2016.

[8] A. Hershkovitz, R. Sitman, R. Israel-Fishelson, A. Eguíluz, P. Garaizar, and M. Guenaga. Creativity in the acquisition of computational thinking. *Interactive Learning Environments*, 27(5-6):628–644, 2019.

[9] R. Israel-Fishelson, A. Hershkovitz, A. Eguíluz, P. Garaizar, and M. Guenaga. The associations between computational thinking and creativity: The role of personal characteristics. *Journal of Educational Computing Research*, 58(8):1415–1447, 2021.

[10] K. H. Kim. Can we trust creativity tests? A review of the torrance tests of creative thinking (TTCT). *Creativity Research Journal*, 18(1):3–14, 2006.

[11] M. Knobelsdorf and R. Romeike. Creativity as a pathway to computer science. In *Proc. ITiCSE*, page 286–290, 2008.

[12] A. Kovalkov, A. Segal, and K. Gal. Inferring creativity in visual programming environments. In *Proc. L@S*, page 269–272, 2020.

[13] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The scratch programming language and environment. *ACM Transactions on Computing Education*, 10(4), Nov. 2010.

[14] D. R. Mullet, A. Willerson, K. N. Lamb, and T. Kettler. Examining teacher perceptions of creativity: A systematic review of the literature. *Thinking Skills and Creativity*, 21:9–30, 2016.

[15] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[16] A. S. B. Reddy and D. S. Juliet. Transfer learning with resnet-50 for malaria cell-image classification. In *Proceedings of the International Conference on Communication and Signal Processing (ICCSP 2019)*, pages 0945–0949, 2019.

[17] M. Resnick, K. Brennan, C. Cobo, and P. Schmidt. Creative learning @ scale. In *Proc. L@S*, page 99–100, 2017.

[18] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[19] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. de Geus. Malicious software classification using transfer learning of resnet-50 deep neural network. In *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA 2017)*, pages 1011–1014, 2017.

[20] M. A. Runco and G. J. Jaeger. The standard definition of creativity. *Creativity Research Journal*, 24(1):92–96, 2012.

[21] R. Shillo, N. Hoernle, and K. Gal. Detecting creativity in an open ended geometry environment. *International Educational Data Mining Society*, 2019.

[22] P. Thomas W., Z. Rui, and B. Tiffany. Evaluation of a data-driven feedback algorithm for open-ended programming. In *Proc. EDM*, pages 192–197, 2017.

[23] E. P. Torrance. Predictive validity of the torrance tests of creative thinking. *The Journal of creative behavior*, 6(4):236–252, 1972.

[24] S. Wheeler, S. Waite, and C. Bromfield. Promoting creative thinking through the use of ict. *Journal of Computer Assisted Learning*, 18(3):367–378, 2002.

[25] F. E. Williams. Assessing creativity across williams "cube" model. *Gifted Child Quarterly*, 23(4):748–756, 1979.

[26] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

ALL-IN-ONE: Multi-Task Learning BERT models for Evaluating Peer Assessments

Qinjin Jia, Jialin Cui, Yunkai Xiao, Chengyuan Liu, Parvez Rashid, Edward Gehringer
Department of Computer Science
North Carolina State University
Raleigh, NC, USA
qjia3, jcui9, yxiao28, cliu32, mrashid4, efg@ncsu.edu

ABSTRACT

Peer assessment has been widely applied across diverse academic fields over the last few decades, and has demonstrated its effectiveness. However, the advantages of peer assessment can only be achieved with high-quality peer reviews. Previous studies have found that high-quality review comments usually comprise several features (e.g., contain suggestions, mention problems, use a positive tone). Thus, researchers have attempted to evaluate peer-review comments by detecting different features using various machine learning and deep learning models. However, there is no single study that investigates using a multi-task learning (MTL) model to detect multiple features simultaneously. This paper presents two MTL models for evaluating peer-review comments by leveraging the state-of-the-art pre-trained language representation models BERT and DistilBERT. Our results demonstrate that BERT-based models significantly outperform previous GloVe-based methods by around 6% in F1-score on tasks of detecting a single feature, and MTL further improves performance while reducing model size.

Keywords

Peer assessment, peer feedback, automated peer-assessment evaluation, text analytics, educational data mining

1. INTRODUCTION

Peer assessment is a process by which students give feedback on other students' work based on a rubric provided by the instructor [20, 24]. This assessment strategy has been widely applied across diverse academic fields, such as computer science [28], medicine [27], and business [1]. Furthermore, massive open online courses (MOOCs) commonly use peer assessment to provide feedback to students and assign grades. There is abundant literature [7, 24, 25, 11] demonstrating the efficacy of peer assessment. For example, Doubling et al. [7] conducted a meta-analysis of 54 controlled experiments for evaluating the effect of peer assessment across subjects and domains. The results indicate that peer assessment is more effective than teacher assessment, and also remarkably robust across a wide range of contexts [7].

However, low-quality peer reviews are a persistent problem in peer assessment, and considerably weaken the learning effect [17, 22]. The advantages of peer assessment can only be achieved with high-quality peer reviews [14]. This suggests that peer reviews should not be simply transmitted to other students but rather should be vetted in some way. Course staff could check the quality of each review comment[1] and assess its credibility manually, but this is not efficient. Sometimes (e.g., for MOOCs), this is not remotely possible. Therefore, to ensure the quality of peer reviews and the efficiency of evaluating their quality, the peer-assessment platform should be capable of assessing peer reviews automatically. We call this Automated Peer-Review Evaluation.

Previous research has determined that high-quality review comments usually comprise several features [14, 2, 25]. Examples of such features are, "contains suggestions", "mentions problems", "uses a positive tone", "is helpful", "is localized" [14]. Thus, one feasible and promising way to evaluate peer reviews automatically is to adjudicate the quality of each review comment based on whether it comprises the predetermined features, by treating this task as a text classification problem. If a peer-review comment does not contain some of the features, the peer-assessment platform could suggest that the reviewer should revise the review comment to add missing features. Additionally, containing suggestions, mentioning problems, and using a positive tone, are among the most essential features. Thus, we use them for this study.

Previous work for automatically evaluating review comments has focused on tasks that detect a single feature. For example, Xiong and Litman [33] designed sophisticated features and used traditional machine-learning methods for identifying peer-review helpfulness. Zingle et al. [37] utilized different rule-based, machine-learning, and deep-learning methods for detecting suggestions in peer-review comments. However, to the best of our knowledge, no single study exists that investigates using a multi-task learning (MTL) model to detect multiple features simultaneously (as illustrated in Figure 1), albeit extensive research has been carried out on

Qinjin Jia, Jialin Cui, Yunkai Xiao, Chengyuan Liu, Parvez Rashid and Edward Gehringer "ALL-IN-ONE: Multi-Task Learning BERT models for Evaluating Peer Assessments". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 525-532. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

[1]In some peer-assessment systems, reviews are "holistic". In others, including the systems we are studying, each review contains a set of review comments, each comment gives a response to a different criterion in the rubric.

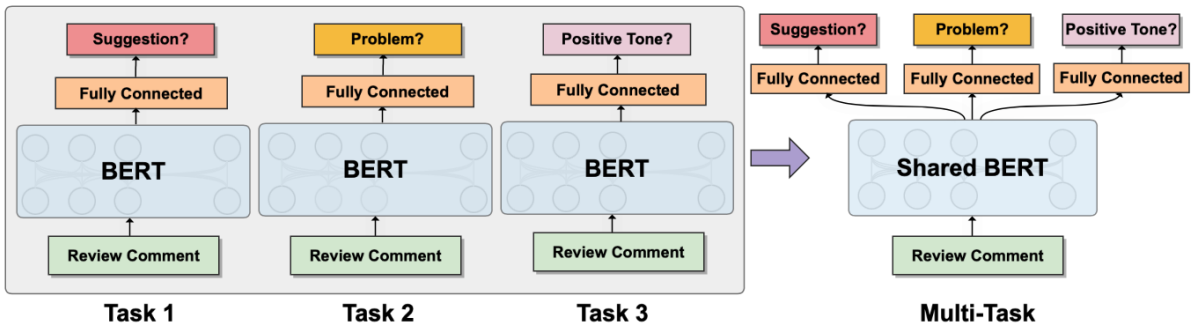


Figure 1: Illustration of the single-task and multi-task learning settings

the topic of automated peer-review evaluation (e.g., [34, 32, 33, 31, 30, 37, 13, 19, 8]).

There are at least two motivations for using multi-task learning (MTL) to detect features simultaneously. Firstly, the problem naturally leads itself well to MTL, due to multiple features usually needing to be employed for a comprehensive and precise evaluation of peer-review comments. If we treat this MTL problem as multiple independent single tasks, total model size and prediction time will increase by a factor of the number of features used for evaluating review comments. Secondly, MTL can increase data efficiency. This implies that learning tasks jointly can lead to performance improvement compared with learning them individually, especially when training samples are limited [5, 36]. More specifically, MTL can be viewed as a form of inductive transfer learning, which can help improve the performance of each jointly learned task by introducing an inductive bias [3].

Additionally, the pre-trained language model, BERT (Bidirectional Encoder Representations from Transformers) [6], has become a standard tool for reaching the state of the art in many natural language processing (NLP) tasks. BERT can significantly reduce the need for labeled data. Therefore, we propose multi-task learning (MTL) models for evaluating review comments by leveraging the state-of-the-art pre-trained language representation models BERT and DistilBERT. We first compare a BERT-based single-task learning (STL) model with the previous GloVe-based STL model. We then propose BERT and DistilBERT based MTL models for jointly learning different tasks simultaneously.

The rest of the paper is organized as follows: Section 2 presents related work. Section 3 describes the dataset used for this study. The proposed single-task and multi-task text classification models are elaborated in Section 4. Section 5 details the experimental setting and results. In Section 6, we conclude the paper, mention the limitations of our research, and discuss future work.

2. RELATED WORK

2.1 Automated Peer-Review Evaluation

The earliest study on automated peer-review evaluation was performed by Cho in 2008 [4]. They manually broke down every peer review comment into review units (self-contained messages in each review comment) and then coded them as praise, criticism, problem detection, solution suggestion. Cho [4] utilized traditional machine learning methods, including naive Bayes, support vector machines (SVM), and decision trees, to classify the review units.

Xiong et al. attempted to use features (e.g., counts of nouns, verbs) derived from regular expressions and dependency parse trees and rule-based methods to detect localization in the review units [32]. Then, they designed more sophisticated features by combining generic linguistic features mined from review comments and specialized features, and used SVM to identify peer-review helpfulness [33]. After that, Xiong et al. upgraded their models to comment-level (use whole review comment instead of review units as the input) [15, 16].

Then, researchers started to use deep neural networks on tasks of automated peer-review evaluation for improving accuracy. Zingle et al. compared rule-based machine-learning and deep neural-network methods for detecting suggestions in peer assessments, and the result showed that deep-learning methods outperformed other traditional methods [37]. Xiao et al. collected around 20,000 peer-review comments and leveraged different neural networks to detect problems in peer assessments [31].

2.2 Multi-Task Learning

Multi-task learning (MTL) is an important subfield of machine learning in which multiple tasks are learned simultaneously [35, 5, 3] to help improve the generalization performance of all the tasks. A task is defined as $\{p(x), p(y|x), L)\}$, where $p(x)$ is the input distribution, $p(y|x)$ is the distribution over the labels given the inputs, and L is the loss function. For the MTL setting in this paper, all tasks have the same input distribution $p(x)$ and loss function L, but different distributions over the labels given the inputs $p(y|x)$.

In the context of deep learning, all methods of MTL can be partitioned into two groups: hard-parameter sharing and soft-parameter sharing [3]. For hard-parameter sharing, the hidden layers are shared between all tasks while keeping several task-specific output layers. For soft-parameter sharing, each task has its independent model, but the distance between the different models' parameters is regularized. For this study, we use the hard-parameter sharing approach.

Table 1: Sample Rubric Criteria

Does the design incorporate all of the functionality required?
Have the authors converted all the cases discussed in the test plan into automated tests?
Does the design appear to be sound, following appropriate principles and using appropriate patterns?

Table 2: Sample Data

Peer-Review Comments (lower-cased)	Sugg.	Prob.	Tone
lots of good background details is given but the testing and implementation sections are missing.	0	1	1
the explanation is clear to follow but it could also include some explanation of the use cases.	1	0	1
only problem statement is explained and nothing about design. please add design and diagrams.	1	1	0

3. DATA

3.1 Data Source: Expertiza[2]

The data in this study is collected from an NSF-funded peer-assessment platform, Expertiza. In this flexible peer-assessment system, students can submit their work and peer-review the learning objects (such as articles, code, and websites) of other students [9]. This platform supports multi-round peer review. In the assignments that provided the review comments for this study, two rounds of peer review (and one round of meta-review) were used:

1. The *formative-feedback* phase: For the first round of review, students upload substantially complete projects. The system then assigns each student to review a set number of these submissions, based on a rubric provided. Sample rubric criteria are provided in Table 1.
2. The *summative-feedback* phase: After students have had an opportunity to revise their work based on feedback from their peers, final deliverables are submitted and peer-reviewed using a summative rubric. The rubric may include criteria such as "How well has the team has addressed the feedback given in the first review round?". Many criteria in the rubric ask reviewers to provide a numeric rating as well as a textual comment.
3. The *meta-review* phase: After the grading period is over, course staff typically assess and grade the reviews provided by students.

For this study, all textual responses to the rubric criteria from the formative-feedback phase and the summative-feedback phase of a graduate-level software-engineering course are extracted to constitute the dataset. Each response to a rubric criterion constitutes a peer-review comment. All responses from one student to a set of criteria in a single rubric are called a peer review or a review. In this study, we focus on evaluating each peer-review comment. After filtering out review comments that only contain symbols and special characters, the dataset consists of 12,053 review comments. In the future, we will update the platform, and this type of review comments will be rejected by the system directly.

3.2 Annotation Process

One annotator who is a fluent English speaker and familiar with the course context annotated the dataset. For quality control, 100 reviews were randomly sampled from the dataset and labeled by a second annotator who is also a fluent English speaker and familiar with the course context. The inter-annotator agreement between two annotators was measured by Cohen's κ coefficient, which is generally thought to be a more robust measure than simple percent agreement calculation [12]. Cohen's κ coefficient for each label is shown in Table 3. The result suggests that the two annotators had almost perfect agreement (>0.81) [12]. Sample annotated comments are provided in Table 2.

[2]https://github.com/expertiza/expertiza

Table 3: Inter-Annotator Agreement (Cohen's κ)

Label	Suggestion	Problem	Tone	Average
Cohen's Kappa	0.92	0.84	0.87	0.88

We define each feature (label) in the context of automated peer-review evaluation as follows:

Suggestion: A comment is said to contain a suggestion if it mentions how to correct a problem or make improvements.

Problem: A comment is said to detect problems if it points out something that is going wrong in peers' work.

Positive Tone: A comment is said to use a positive tone if it has an overall positive semantic orientation.

3.3 Statistics on the Dataset

The minority class for each label includes more than 20% of samples, and thus the dataset is mildly imbalanced. It consists of 12,053 peer-review comments, and the average number of words for each peer-review comment is 29. We found that most students (over three-quarters) use a positive tone in their peer-review comments. Around half of the review comments mention problems with their peers' work, but only one-fifth of review comments give suggestions. Characteristics of the dataset are shown in Table 4 below,

Table 4: Statistics on the Dataset

Label	Class	%samples	avg.#words	max#words
Sugg.	0	79.2%	22	922
	1	20.8%	58	1076
Prob.	0	56.7%	22	479
	1	43.3%	38	1076
Pos. Tone	0	22.2%	28	1040
	1	77.8%	29	1076

4. METHODOLOGY

In this section, we first briefly introduce Transformer [26], BERT [6], and DistilBERT [21]. Then we describe BERT and DistilBERT based single-task and multi-task models.

4.1 Transformer

In 2017, Vaswani et al. published a groundbreaking paper, "Attention is all you need," and proposed an architecture called Transformer, which significantly improved the performance of sequence-to-sequence tasks (e.g., machine translation) [26]. The Transformer is entirely built upon self-attention mechanisms without using any recurrent or convolutional layers. As shown in Figure 2, the Transformer consists of two parts: the left part is an *encoder*, and the right part is a *decoder*. The *encoder* block takes a batch of sentences represented as sequences of word IDs. Then the sequences pass through an embedding layer, and the positional embedding adds positional information of each word.

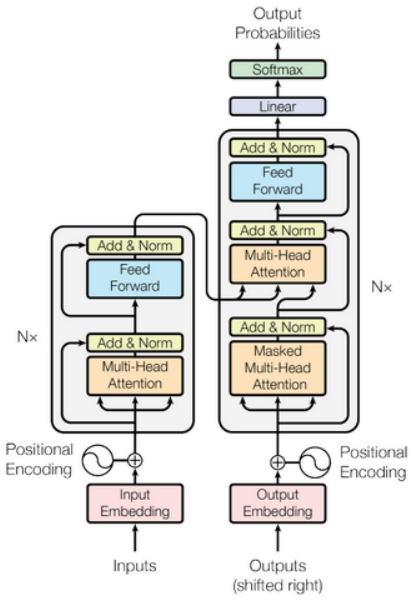


Figure 2: Architecture of the Transformer [26]

The *encoder* block is then briefly introduced since BERT reuses it. Each *encoder* consists of two layers: a multi-head attention layer and a feed-forward layer. The multi-head attention layer uses the self-attention mechanism, which encodes each word's relationship with every other word in the same sequence, paying more attention to the most relevant ones. For example, the output of this layer for the word "like" in the sentence, "we like the Educational Data Mining conference 2021!" will depend on all the words in the sentence. However, it will probably pay more attention to the word "we" than to the words "data" or "mining."

4.2 BERT

BERT is a state-of-the-art pre-trained language representation model proposed by Devlin et al. [6]. It has advanced the state-of-the-art results in many NLP tasks and significantly reduced the need for labeled data by pre-training on unlabeled data over different pre-training tasks. Each BERT model consists of 12 *encoder* blocks of the Transformer model. The input representation is constructed by summing the corresponding token and positional embeddings. The length of the output sequence is the same as the input length, and each input token has a corresponding representation in the output. The output of the first token '[CLS]' (a special token added to the sequence) is utilized as the aggregate representation of the input sequence for classification tasks [6].

The BERT framework consists of two steps: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data, BooksCorpus (800M words) and English Wikipedia (2,500M words), over two pre-training tasks, Masked language model (MLM) and Next sentence prediction (NSP). For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and then all of the parameters are fine-tuned using labeled data from the downstream tasks (e.g., text classification). For this study, we use HuggingFace pre-trained BERT[3] to initialize models and then fine-tune models with annotated peer-review comments for automated peer-review evaluation tasks.

4.3 DistilBERT

Although BERT has shown remarkable improvements across various NLP tasks and can be easily fine-tuned for downstream tasks, one main drawback of BERT is that it is very compute-intensive (i.e., it takes a huge amount of parameters, ∼110M parameters). Therefore, researchers are attempting to apply different methods for compressing BERT, including pruning, quantization, and knowledge distillation [10]. One of the compressed BERT models is called DistilBERT [21]. DistilBERT is compressed from BERT by leveraging the knowledge distillation technique during the pre-training phase. The authors [21] demonstrated that DistilBERT has 40% fewer parameters and is 60% faster than the original BERT while retaining 97% of its language-understanding capabilities. We will investigate whether we can reduce model size while retaining performance for our task with DisilBERT.[4]

4.4 Input Preparation

Text Preprocessing: First, URL links in peer-review comments are removed. Then, we lowercase all comments and leverage a spellchecker API[5] to correct typos and misspellings. Finally, two special tokens ([CLS], [SEP]) are added to each review comment, as required for BERT. The [CLS] token is added to the beginning of each review for classification tasks. The [SEP] token is added at the end of each review.

Subword Tokenization: The tokenizer used for BERT is a subword tokenizer called "WordPiece" [29]. Traditional word tokenizers suffer the out-of-vocabulary (OOV) word problem. However, a subword tokenizer could alleviate the OOV problem. It splits a text into subwords, which then are converted to token IDs.

Input Representation: The token IDs are padded or truncated to 100 for each sequence and then pass through a trainable embedding layer to be converted to token embeddings. The input representation for BERT is constructed by summing the token embeddings and positional embeddings.

[3]https://huggingface.co/bert-base-uncased
[4]https://huggingface.co/distilbert-base-uncased
[5]https://pypi.org/project/pyspellchecker/

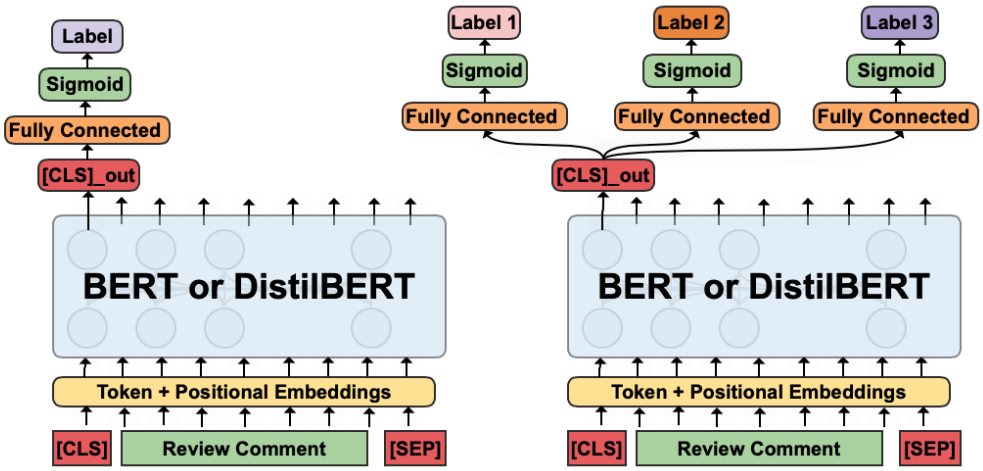


Figure 3: BERT and DistilBERT based single-task and multi-task learning architectures

4.5 Single-Task and Multi-Task Models

As mentioned in the BERT paper [6] and other studies [23], the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, including text classification. Therefore, only one dense layer is added on top of the original BERT or DistilBERT model and used as a binary classifier for the single-task learning models. Three dense layers are added to the multi-task learning models, one for each label.

5. EXPERIMENTS AND RESULTS

In this section, we first introduce training details and evaluation metrics and then show experimental results.

5.1 Training

Train/Test split: We find by experiments that increasing training size does not help the classifier when the number of training samples is over 5000. Therefore, 5000/2053/5000 data samples are used for training/validation/testing.

Loss Functions: For BERT and DistilBERT based Single-Task Learning (STL) models, the cross-entropy loss is used. For BERT and DistilBERT base Multi-Task Learning (MTL) models, the cross-entropy loss is used for each task. The total loss will be the sum of the cross-entropy loss of each task.

Cost-Sensitive method: As mentioned in Section 3.3, the dataset is mildly imbalanced (minority class $> 20\%$). Thus, a cost-sensitive method is used in this study for alleviating the problem of class imbalance and improving performance, by weighting the cross-entropy loss function during training based on the frequency of each class in the training set.

Hyperparameters: As we mentioned in Section 4.2 and Section 4.3, we use HuggingFace pre-trained BERT and DistilBERT to initialize models. The hidden size for BERT and DistilBERT is 768. We then fine-tune the BERT and DistilBERT based single-task learning and multi-task learning models with a batch size of 32, max sequence length 100, learning rate 2e-5/3e-5/5e-5, epochs of 2/3, dropout rate 0.1, and Adam optimizer with β_1=0.9 and β_2=0.99.

5.2 Evaluation Metrics

We use accuracy, macro-F1 score (average for each class of each label instead of each label), and AUC (Area Under ROC Curve) to evaluate models. Since the dataset is merely mildly imbalanced, accuracy can still be a useful metric. The Macro-F1 instead of F1-score for the positive class is used, since both positive class and negative class for each label are important for our task. For this study, we mainly use accuracy and macro-F1 to compare different models.

5.3 Results

Table 5 shows the performance of all models when training with a different number of training samples (1K, 3K, and 5K). The first column indicates the models (GloVe, BERT, DistilBERT) and training settings (single-task learning (STL), multi-task learning (MTL)).

RQ1 Does BERT outperform previous methods?
We first implemented a baseline single-task learning model by leveraging pre-trained GloVe (Global Vectors for Word Representation)[6] [18] word embeddings. We added a BatchNormalization layer on top of GloVe, and the aggregate representation of the input sequence for classification was obtained by AveragePooling the output of the BatchNormalization layer. A dense layer was added on the top for performing classification.

We compared GloVe and BERT for every single task. As shown in Table 5, the results clearly showed that a BERT-based STL model yields substantial improvements over the previous GloVe-based method. The STL-BERT model trained with 1000 data samples outperformed the STL-GloVe model trained with 5000 data samples on all tasks. This suggests that the need for labeled data could be significantly reduced by leveraging a pre-trained language model BERT.

RQ2 How does multi-task learning perform?
By comparing MTL-BERT with STL-BERT and MTL-DistilBERT with STL-DistilBERT when trained with a different number of training samples, we found that jointly learning

[6]https://nlp.stanford.edu/projects/glove/

Table 5: Performance evaluation (average performance of 5 independent runs)

	Suggestion			Problem			Pos. Tone		
	Acc.	Macro-F1	AUC	Acc.	Macro-F1	AUC	Acc.	Macro-F1	AUC
Training with 1000 labeled data samples									
1 STL-GloVe (Baseline)	82.0%	.744	.865	80.2%	.790	.879	76.0%	.700	.823
2 STL-BERT	90.0%	.868	**.975**	**89.2%**	**.892**	**.955**	87.0%	.828	.940
3 MTL-BERT	**94.0%**	.904	.974	89.0%	.890	**.955**	**89.4%**	**.846**	**.941**
4 STL-DistilBERT	92.4%	.890	.970	88.0%	.880	.950	86.2%	.822	.933
5 MTL-DistilBERT	93.8%	**.910**	.971	89.0%	.886	.951	88.6%	.824	.939
Training with 3000 labeled data samples									
1 STL-GloVe (Baseline)	88.4%	.836	.929	83.0%	.830	.898	82.4%	.770	.872
2 STL-BERT	93.8%	.910	.980	90.6%	.904	**.964**	89.6%	.858	**.948**
3 MTL-BERT	**94.6%**	**.916**	**.981**	**91.0%**	**.906**	**.964**	90.0%	**.854**	.947
4 STL-DistilBERT	94.0%	.910	.979	89.8%	.900	.962	89.0%	.850	.942
5 MTL-DistilBERT	94.2%	**.916**	.978	89.6%	.892	.960	**90.2%**	.850	.945
Training with 5000 labeled data samples									
1 STL-GloVe (Baseline)	89.9%	.852	.947	84.2%	.832	.908	85.0%	.794	.883
2 STL-BERT	94.4%	.916	.980	**91.2%**	**.912**	**.968**	89.4%	.852	.950
3 MTL-BERT	**94.8%**	**.922**	**.982**	91.0%	.908	.966	**90.8%**	**.854**	**.951**
4 STL-DistilBERT	94.2%	.912	.978	90.4%	.902	.964	89.8%	.860	.944
5 MTL-DistilBERT	94.2%	.914	.980	90.4%	.902	.964	90.6%	.852	**.951**

Table 6: The # of parameters for each setting

Setting	# of parameters
STL-BERT * 3	328M
STL-DistilBERT * 3	199M
MTL-BERT	109M
MTL-DistilBERT	66M

related tasks improves the performance of the suggestion-detection task and the positive-tone detection task, especially when we have limited training samples (i.e., when training with 1K and 3K data samples). This suggests that MTL can increase data efficiency. However, for the problem-detection task, there is no significant difference between the performance of the STL and MTL settings.

Additionally, MTL can considerably reduce the model size. As shown in Table 6, three BERT-based STL models would have more than 328M parameters, and this number would be 199M for the DistilBert-based models. However, if we employ the MTL models for evaluating peer-review comments, the number of parameters would be reduced to 109M and 66M, respectively. This demonstrates that using MTL to evaluate reviews can save considerable memory resources and reduce the response time of peer-review platforms.

RQ3 How does DistilBERT perform?
By comparing DistilBERT and BERT on both STL and MTL settings, we found that BERT-based models slightly outperformed DistilBERT-based models. This result implied a trade-off between performance and model size when selecting the model to be deployed on peer-review platforms. If we focus on high accuracy instead of memory resource usage and response time of the platforms, the MTL-BERT model is the choice. Otherwise, the MTL-DistilBERT should be deployed.

6. CONCLUSIONS

In this study, we implemented single-task and multi-task models for evaluating peer-review comments based on the state-of-the-art language representation models BERT and DistilBERT. Overall, the results showed that BERT-based STL models yield significant improvements over the previous GloVe-based method on tasks of detecting a single feature. Jointly learning different tasks simultaneously further improves performance and saves considerable memory usage and response time for peer-review platforms. The MTL-BERT model should be deployed on peer-review platforms, if our focus is on high accuracy instead of memory resource usage and response time of the platforms. Otherwise, the MTL-DistilBERT model is preferred.

There are three limitations to this study. Firstly, we employed three features of high-quality peer reviews to evaluate a peer-review comment. However, it is still unclear how MTL will perform if we learn more tasks simultaneously. Secondly, we mainly focused on a hard-parameter sharing approach for constructing MTL models. However, some studies have found that the soft-parameter sharing approach might be a more effective method for constructing multi-task learning models. Thirdly, the performance of the model has not been evaluated in actual classes. We intend to deploy the model on the peer-review platform and evaluate the model extrinsically in real-world circumstances.

These preliminary results serve as a basis for our ongoing work, in which we are building a more complex all-in-one model for comprehensively and automatically evaluating the quality of peer review comments to improve peer assessment. In the future, we will attempt to evaluate peer reviews based on more predetermined features and use fine-grained labels (e.g., instead of evaluating whether a peer-review comment contains suggestions, we will evaluate how many suggestions are contained in a review comment).

7. REFERENCES

[1] S. Brutus, M. B. Donia, and S. Ronen. Can business students learn to evaluate better? evidence from repeated exposure to a peer-evaluation system. *Academy of Management Learning & Education*, 12(1):18–31, 2013.

[2] P. Caligiuri and D. C. Thomas. From the editors: How to write a high-quality review, 2013.

[3] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[4] K. Cho. Machine classification of peer comments in physics. In *Educational Data Mining 2008*, 2008.

[5] M. Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] K. S. Double, J. A. McGrane, and T. N. Hopfenbeck. The impact of peer assessment on academic performance: A meta-analysis of control group studies, 2020.

[8] M. Fromm, E. Faerman, M. Berrendorf, S. Bhargava, R. Qi, Y. Zhang, L. Dennert, S. Selle, Y. Mao, and T. Seidl. Argument mining driven analysis of peer-reviews. *arXiv preprint arXiv:2012.07743*, 2020.

[9] E. Gehringer, L. Ehresman, S. G. Conger, and P. Wagle. Reusable learning objects through peer review: The expertiza approach. *Innovate: Journal of Online Education*, 3(5):4, 2007.

[10] M. Gupta, V. Varma, S. Damani, and K. N. Narahari. Compression of deep learning models for nlp. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3507–3508, 2020.

[11] H. Li, Y. Xiong, C. V. Hunter, X. Guo, and R. Tywoniw. Does peer assessment promote student learning? a meta-analysis. *Assessment & Evaluation in Higher Education*, 45(2):193–211, 2020.

[12] M. L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282, 2012.

[13] S. Negi and P. Buitelaar. Suggestion mining from opinionated text. *Sentiment Analysis in Social Networks*, pages 129–139, 2017.

[14] M. M. Nelson and C. D. Schunn. The nature of feedback: How different types of peer feedback affect writing performance. *Instructional Science*, 37(4):375–401, 2009.

[15] H. Nguyen, W. Xiong, and D. Litman. Classroom evaluation of a scaffolding intervention for improving peer review localization. In *International Conference on Intelligent Tutoring Systems*, pages 272–282. Springer, 2014.

[16] H. Nguyen, W. Xiong, and D. Litman. Instant feedback for increasing the presence of solutions in peer reviews. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 6–10, 2016.

[17] D. Omar, M. Shahrill, and M. Zuraifah Sajali. The use of peer assessment to improve students' learning of geometry. *European Journal of Social Science Education and Research*, 5(2):187–206, 2018.

[18] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[19] L. Ramachandran et al. Automated assessment of reviews. 2013.

[20] P. M. Sadler and E. Good. The impact of self-and peer-grading on student learning. *Educational assessment*, 11(1):1–31, 2006.

[21] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[22] H. K. Suen. Peer assessment for massive open online courses (moocs). *International Review of Research in Open and Distributed Learning*, 15(3):312–327, 2014.

[23] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? *arXiv preprint arXiv:1905.05583*, 2019.

[24] K. Topping. Peer assessment between students in colleges and universities. *Review of educational Research*, 68(3):249–276, 1998.

[25] M. Van Zundert, D. Sluijsmans, and J. Van Merriënboer. Effective peer assessment processes: Research findings and future directions. *Learning and instruction*, 20(4):270–279, 2010.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[27] C. Violato and J. Lockyer. Self and peer assessment of pediatricians, psychiatrists and medicine specialists: implications for self-directed learning. *Advances in Health Sciences Education*, 11(3):235–244, 2006.

[28] Y. Wang, H. Li, Y. Feng, Y. Jiang, and Y. Liu. Assessment of programming language learning based on peer code review model: Implementation and experience report. *Computers & Education*, 59(2):412–422, 2012.

[29] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[30] Y. Xiao, G. Zingle, Q. Jia, S. Akbar, Y. Song, M. Dong, L. Qi, and E. Gehringer. Problem detection in peer assessments between subjects by effective transfer learning and active learning.

[31] Y. Xiao, G. Zingle, Q. Jia, H. R. Shah, Y. Zhang, T. Li, M. Karovaliya, W. Zhao, Y. Song, J. Ji, et al. Detecting problem statements in peer assessments. *arXiv preprint arXiv:2006.04532*, 2020.

[32] W. Xiong and D. Litman. Identifying problem localization in peer-review feedback. In *International Conference on Intelligent Tutoring Systems*, pages 429–431. Springer, 2010.

[33] W. Xiong and D. Litman. Automatically predicting

peer-review helpfulness. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 502–507, 2011.

[34] W. Xiong, D. Litman, and C. Schunn. Assessing reviewers' performance based on mining problem localization in peer-review data. In *Educational Data Mining 2010-3rd International Conference on Educational Data Mining*, pages 211–220, 2010.

[35] Y. Zhang and Q. Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[36] Y. Zhang and Q. Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[37] G. Zingle, B. Radhakrishnan, Y. Xiao, E. Gehringer, Z. Xiao, F. Pramudianto, G. Khurana, and A. Arnav. Detecting suggestions in peer assessments. *International Educational Data Mining Society*, 2019.

Quizzing Policy Using Reinforcement Learning for Inferring the Student Knowledge State

Joy He-Yueya
University of Washington
joyhe@cs.washington.edu

Adish Singla
MPI-SWS
adishs@mpi-sws.org

ABSTRACT

The prevalence of online education systems provides opportunities to deliver personalized learning at scale. Educational systems need to assess students so that they can provide better curricula tailored to each student's unique needs. Since there is a limited amount of time for quizzing a student, we need to test each student using those questions that capture the most information about their level of understanding of various concepts. In this paper, we formally pose the problem and present multiple approaches for learning a quizzing policy to determine a personalized sequence of questions for each student that best predicts their knowledge state. We first introduce simple heuristics including random selection and an uncertainty sampling approach inspired by an active learning framework. We then develop a reinforcement learning (RL) approach for designing a quizzing policy. Using simulations of students' knowledge states, we provide initial evidence that an RL-based approach can improve over simple heuristics. We further demonstrate the effectiveness of our approaches using a real-world dataset consisting of over 1.5 million examples of students' answers to mathematics questions from Eedi, an online educational platform.

Keywords

reinforcement learning, knowledge state, quizzing policy

1. INTRODUCTION

Online education systems are making high-quality education more accessible for students across the globe. These systems provide various educational resources such as instructional videos and exercises. To provide personalized curricula for improving the learning outcomes of students, an online education system needs to accurately infer each student's knowledge state (i.e., their level of understanding of various concepts) by quizzing them. This is a challenging task because the quizzing time is limited. To make the most efficient use of each student's time, it is important to prioritize those questions that reveal the most information about the student's knowledge.

We focus on a specific goal for student assessment: *given a limit to the number questions we are allowed to ask each student, how can we determine a sequence of questions for each student that best predicts their knowledge state?* Specifically, when an education system needs to assess a student for inferring their knowledge, the system suggests a personalized question to query for the student and gets their response to the question. Based on the student's response history (i.e., a sequence of question-response pairs), the system selects another question to query for the student until it has exhausted its query budget (i.e., the maximum number of queries allowed). We refer to the function that provides the next question to query based on students' response histories as quizzing policy (QP).

We define the task of learning a QP in the context of the NeurIPS 2020 Education Challenge [27] launched by Eedi [6], an online educational platform with thousands of active users daily around the globe. We consider a set of 948 multiple-choice mathematics questions that correspond to 57 different concepts. Specifically, the task is to obtain a limited set of answers from each student for inferring the student's knowledge on the 57 concepts and then predict the student's performance on unseen questions based on the inferred knowledge state.

The key challenge in designing a QP is related to a crucial task in machine learning: active learning (AL). For many learning tasks (e.g., image classification, text classification), obtaining sufficient labeled data for training high-performance models is costly [16, 18, 32]. AL aims to reduce the amount of annotated data needed by having the model carefully select which data points should be labeled.

Existing methods for AL include heuristics such as selecting the data points about which the model is most uncertain (i.e., uncertainty sampling) [15, 26, 31, 24], picking the instances about which a set of possible different models disagree the most (i.e., query by committee) [23, 10], or choosing the example that can lead to the most immediate improvement in model performance (i.e., estimated error reduction) [22, 12].

In addition to these heuristics for AL, recent studies [30, 19, 9] have explored how to use reinforcement learning (RL) to learn the AL strategy itself. RL [20, 25] is a powerful

Joy He-Yueya and Adish Singla "Quizzing Policy Using Reinforcement Learning for Inferring the Student Knowledge State". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 533-539. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

framework where an agent learns how to make good decisions (actions) in different situations (states) through trial and error. In the RL terminology, the action space provides the set of actions that can be taken by the RL agent at a given point in time; the state space defines the "state of the world" that is visible to the RL agent; and the reward function assigns a value to the outcome of each action taken by the RL agent. In this case, the set of possible instances to be labeled defines the action space; the state space is a representation of the sequence of instances that have already been annotated; and the gain in prediction accuracy as a result of an action defines the reward. The RL agent learns to improve its decision-making over time based on the reward signals it receives. Inspired by these studies, we investigate using RL to learn a QP for personalized student assessment.

1.1 Our approach and contributions

In this paper, we formalize the problem of learning a QP for inferring the student knowledge state and present several different approaches including simple heuristics and an RL-based approach. Our contributions are:

- We formulate the problem of learning a QP to infer student knowledge.
- We propose simple heuristics (i.e., random selection, uncertainty sampling) and an RL-based approach for learning a QP.
- We evaluate the performance of different QPs on a synthetic dataset and a publicly available dataset consisting of over 1.5 million examples of students' answers to mathematics questions from Eedi.

For the reproducibility of experimental results and facilitating research in this area, the code and dataset are publicly available.[1]

1.2 Related work

AL is a popular methodology in machine learning that aims to reduce the amount of annotated data needed by having the model carefully select which data points should be labeled. The task of designing a QP is closely related to AL because the goal is to optimally select a set of questions to ask students to gain the most information about their knowledge states. Uncertainty sampling [15, 26, 31, 24] is one of the most popular heuristics for AL because it is straightforward and computationally efficient. Specifically, it suggests labeling instances that are closest to the model's decision boundary (i.e., the most uncertain). Woodward and Finn [30] propose the first application of RL to the task of AL for image classification. Other studies [19, 9] explore how to train an AL policy that can generalize across diverse datasets.

RL has also been applied to various tasks in education such as learning an instructional policy [2, 3, 5, 13, 17, 21, 28], learning a hint policy for helping students solve multi-step problems [7], and generating new educational tasks [1]. We introduce a different policy, a quizzing policy for inferring the student knowledge state, which has not been designed using RL in previous literature.

[1]https://github.com/joyheyueya/quizzing-policy

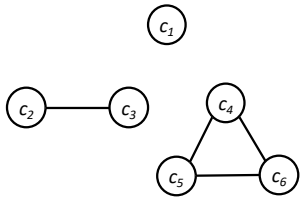


Figure 1: Graphical representation of knowledge. This is an example of an undirected graph where each node (circle) represents a concept, and each edge connects a pair of similar concepts: c_1 is an independent concept, c_2 and c_3 are similar, and c_4, c_5, and c_6 are similar.

There is prior work on the efficient assessment of knowledge [8]. Our student knowledge model is inspired by the knowledge components (i.e., concepts / skills) used in Bayesian Knowledge Tracing (BKT) [4], which represents the state for each knowledge component as a binary variable: 1 if the knowledge component is known, 0 otherwise.

2. PROBLEM FORMULATION

In this section, we formalize the problem of learning a quizzing policy (QP) for inferring the student knowledge state.

2.1 Student knowledge state

Our goal is to infer student knowledge on a set of n concepts $C = \{c_1, ..., c_n\}$ associated with a set of m questions $X = \{x_1, ..., x_m\}$. For simplicity, each question corresponds to a single concept, but each concept might be associated with more than one question ($m >> n$). A student's knowledge state h is defined as $h = [v_1, ..., v_n]$ where $v_1, ..., v_n$ are binary variables that indicate whether or not the student knows each concept in C: $v_i = 1$ if c_i is known, and $v_i = 0$ otherwise. Formally, we define a hypothesis space $\mathcal{H}$ for all possible knowledge states: $\mathcal{H} = \{0, 1\}^n$. We assume h is fixed during the assessment.

2.2 Graphical representation of knowledge

We consider two assumptions that are useful for inferring the student knowledge state: 1) difficult concepts are more likely to be unknown, and easy concepts are more likely to be known; 2) similar concepts are more likely to have the same value (i.e., a student who knows one concept is also likely to know the other concepts that are similar to the one that is already known). These influences can be represented by an undirected graph where each node corresponds to a concept, and each edge connects a pair of concepts that are similar (see Figure 1). In the Eedi dataset (described in Section 4.2.1), we consider every pair of concepts that share the same super-concept to be similar (e.g., there is an edge between "Rearranging Formula and Equations" and "Substitution into Formula" because they are both under the same super-concept "Formula"). Based on this graphical structure, we model a student's knowledge state using a Markov Random Field (MRF).

An MRF is a probability distribution over a set of variables that satisfy certain properties defined by an undirected graph. In our case, we define a probability distribution p over binary variables $v_1, ..., v_n$ defined by an undirected graph $G = (V \cup F, E)$ where V is the set of nodes (con-

cepts), F is the set of factors that define a set of functions over the variables that they are connected with, and E is the set of edges (see Figure 2).

An MRF allows us to calculate the probability of each way of assigning values to binary variables $v_1, ...v_n$, which represent the knowledge state of the corresponding concepts $c_1, ..., c_n$. The probability p has the form:

$$p(v_1, ..., v_n) = \frac{1}{Z} \prod_{\psi_\alpha \in F} \psi_\alpha(v_\alpha) \quad (1)$$

where α represents a subgraph of G, and ψ_α denotes a factor that defines a non-negative function over the set of variables v_α in α. Z is a normalizing constant that ensures the distribution sums to one:

$$Z = \sum_{v_1,...v_n} \prod_{\psi_\alpha \in F} \psi_\alpha(v_\alpha) \quad (2)$$

We specify factors for an MRF based on two assumptions about variables $v_1, ..., v_n$. For our first assumption that difficult concepts have a higher probability of being unknown, we define unary factors:

$$\psi_i(v_i) = \begin{cases} 1 - \textit{difficulty}_{c_i} & \text{if } v_i = 1 \\ \textit{difficulty}_{c_i} & \text{otherwise} \end{cases}$$

where $\textit{difficulty}_{c_i}$ is a real number that represents the difficulty of the concept c_i that v_i corresponds to, and $0 \leq \textit{difficulty}_{c_i} \leq 1$. A higher $\textit{difficulty}_{c_i}$ value means c_i is more difficult.

For our second assumption that variables corresponding to similar concepts are more likely to have the same values, we define binary factors between every pair of nodes (v_i, v_j) that are connected by an edge in graph G:

$$\psi_{\{i,j\}}(v_i, v_j) = \begin{cases} \textit{influence} & \text{if } v_i = v_j \\ 1 - \textit{influence} & \text{otherwise} \end{cases}$$

where *influence* represents a constant that satisfies $0.5 \leq \textit{influence} \leq 1$. A greater *influence* value means we want to assign a higher probability to an assignment that gives the same values to variables corresponding to similar concepts. In our work, we fix *influence* to be 0.7. We also tried similar values, and they lead to similar results.

2.3 Quizzing policy for knowledge inference

Since there is a cost associated with each question we query students (e.g., time, student's energy), we need to select a limited number of questions that reveal the most about their knowledge state. Thus, student knowledge prediction can be framed as a pool-based active learning (AL) task with a given query budget T. For simplicity, we assume querying each exercise leads to the same cost and define T to be the total number of queries we are allowed.

We describe the AL framework in detail, see Algorithm 1. At a given time step t, we have a labelled set L that consists of all the questions we have asked the student and their responses. Formally, $L = \{(x^i, y^i)\}_{i=1}^t$ where $x^i \in X$, and $y^i \in \{0, 1\}$ is the student's response to x^i ($y^i = 0$ if the response is incorrect, $y^i = 1$ if the response is correct). We also have an unlabelled set U consisting of all the questions that we have not asked). Based on L, we have a belief B_h about the student's knowledge h. Formally, $B_h = [b_1, ..., b_n]$ where b_i is the probability of knowing the concept c_i (i.e., $v_i = 1$ with a probability of b_i). We define $Binary(B_h)$ as a function that converts probabilities into binary values using a threshold of 0.5 (1 if $b_i \geq 0.5$ and 0 otherwise). $Binary(B_h)$ gives the inferred binary knowledge state. We update B_h based on L by running the Loopy Belief Propagation algorithm (LBP) [11] on our graph defined in Section 2.2. LBP takes L as input and outputs the probabilities $b_1, ..., b_n$ ($0 \leq b_i \leq 1$). Additionally, we have a QP that takes B_h as input and outputs the next question to ask the student. Specifically, a policy $\pi(\cdot|B_h)$ provides a probability distribution with support over all questions in U given B_h. We can then sample a question from $\pi(\cdot|B_h)$.

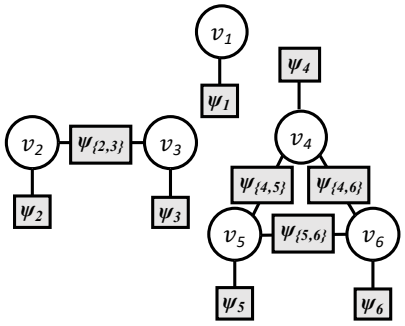


Figure 2: Modeling graphical student knowledge using MRF. This models the knowledge representation in Figure 1 as a factor graph. Each node v_i is a binary variable that represents the knowledge state of the corresponding concept c_i. Factors are represented by rectangles. There is a unary factor for every node and a binary factor between every pair of nodes connected by an edge to model the dependency between variables.

Algorithm 1: Active learning for inferring knowledge

Input: budget T, quizzing policy π
Output: $\hat{h}$
Initialize $L_0 \leftarrow \emptyset$, $U_0 \leftarrow \{x_i\}_{i=1}^m$
for $t = 1, 2, 3, \ldots, T$ **do**
  $B_{h_t} = \text{LBP}(L_{t-1})$
  $x^t \sim \pi(\cdot|B_{h_t})$
  $L_t \leftarrow L_{t-1} \cup (x^t, y^t)$
  $U_t \leftarrow U_{t-1} \backslash x^t$
end
$\hat{h} \leftarrow Binary(B_{h_T})$

Algorithm 1 runs as follows: at each time step t, we first get our current belief B_{h_t} based on the previously labelled set L_{t-1} (i.e., the set of all the questions we have asked the student before time step t and their responses). We then select a question x^t from the previously unlabelled set U_{t-1} to ask the student by sampling from $\pi(\cdot|B_{h_t})$, which defines a probability distribution with support over all questions in U_{t-1} given B_{h_t}. Then, we update U_{t-1} to U_t by removing x^t from U_{t-1} and update L_{t-1} to L_t by adding x^t and its label y^t to L_{t-1}. The quizzing process terminates when the query budget is exhausted. In this work, we fix $T = 10$ as required

by the NeurIPS 2020 Education Challenge [27]. The final output of the algorithm $\hat{h}$ is the student's knowledge state at time step T, which is inferred based on B_{h_T}.

2.4 Evaluation

We evaluate our QPs using two methods. First, we create a synthetic dataset consisting of simulated students (see Section 4.1). We predict each student's knowledge state using Algorithm 1. Given a prediction result $\hat{h}$, we calculate the prediction accuracy using the following equation:

$$Acc(\hat{h}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\hat{h}[i] = h^{\star}[i]) \tag{3}$$

where $h^{\star}$ is the actual knowledge state.

Second, we apply our QPs to the NeurIPS 2020 Education Challenge (see Section 4.2). The challenge is to obtain a limited set of answers from each student for predicting the correctness of their answers to the remaining questions. Our approach to this challenge is to first infer a student's knowledge state using Algorithm 1 and then predict the student's responses to the remaining questions based on the inferred knowledge state. Specifically, we design an additional model (see Section 4.2.2) that takes in our belief about the student's knowledge state at the final time step B_{h_T} and outputs the student's response to each of the m questions. Formally, the vector $\hat{\mathcal{Y}} \in \mathbb{R}^m$ denotes the output of the model. We calculate the prediction accuracy as:

$$Acc(\hat{\mathcal{Y}}) = \frac{1}{|U_T|}\sum_{x_i \in U_T} \mathbb{1}(\hat{\mathcal{Y}}[i] = \mathcal{Y}^{\star}[i]) \tag{4}$$

where U_T is the set of unlabelled questions at the final time step (unseen by the model), $\mathcal{Y}^{\star}[i]$ is the student's actual response to x_i, and $\hat{\mathcal{Y}}[i]$ is the predicted response.

3. DESIGNING QUIZZING POLICIES

In this section, we present heuristics-based approaches and a reinforcement learning (RL)-based approach to designing a quizzing policy (QP) that takes in a belief B_h about a student's knowledge state and outputs the next question to ask the student.

3.1 Heuristic approaches

We present two simple heuristics for designing a QP: random selection (QP-Random) and uncertainty sampling (QP-Uncertain). QP-Random is straightforward: we always randomly select a question from the unlabelled set U (i.e., $\pi(a|B_h) = \frac{1}{|U|}$ for each $a \in U$). QP-Uncertain suggests picking a question corresponding to a concept that our current model is most uncertain about (i.e., the concept with a probability of being known that is closest to 0.5). Formally, we define:

$$b^{\star} = \arg\min_{b_i \in B_h} |b_i - 0.5|$$

We first pick a concept $c^{\star}$ with a probability of being known that is equal to $b^{\star}$. We break ties randomly. We define $U_{c^{\star}}$ as the set of questions that have not been asked and are associated with $c^{\star}$. We then define the policy:

$$\pi(a|B_h) = \begin{cases} \frac{1}{|U_{c^{\star}}|} & \text{if } a \in U_{c^{\star}} \\ 0 & \text{otherwise} \end{cases}$$

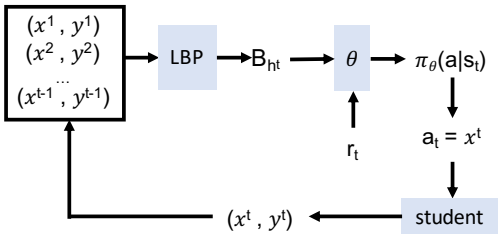


Figure 3: QP-RL approach.

3.2 RL-based approach

We now propose an RL-based approach (QP-RL) for learning a QP. An RL agent learns how to make good decisions over time by interacting with an environment that is typically modeled as a Markov Decision Process (MDP). In our problem setting, we define the MDP $M = (S, A, P, R, s_0)$ as follows:

- The state space S is the set of beliefs B_h about student knowledge (i.e., $S = \{[b_1, ..., b_n] | 0 \leq b_i \leq 1\}$);
- The action space A is the set of questions that have not been asked;
- The transition dynamics $P : S \times A \times S \to \mathbb{R}$ define the probability of transitioning from one state to another by taking a particular action. In our case, we transition to state s_{t+1} from s_t based on the student's response y^t.
- The reward function $R : S \times A \times S \to \mathbb{R}$ is defined as the difference in prediction accuracy between the current time step and previous step: for predicting student knowledge, given the inferred knowledge state h_{t+1} after taking action a_t, we calculate the reward for time step t as $Acc(h_{t+1}) - Acc(h_t)$;
- The initial state s_0 corresponds to the initial belief about student knowledge: each concept has a 0.5 probability of being known.

Figure 3 shows an overview of the QP-RL approach. For training the RL agent, we consider an episodic, finite-horizon setting. During each episode, we train on one student's data, and the length of the episode is the query budget T. At each time step t, we run the LBP algorithm that takes in the student's response history $L_{t-1} = \{(x^i, y^i)\}_{i=1}^{t-1}$ to update our belief about the student's knowledge state B_{h_t}. Then, the RL model, which is a neural network with parameters θ, takes B_{h_t} as input (i.e., $s_t = B_{h_t}$) and outputs a vector $\mathbf{p_c} \in \mathbb{R}^n$ which represents the probability of selecting a question corresponding to each of the n concepts. We first select a concept c_i by sampling based on $\mathbf{p_c}$ and then randomly select one question from U_{c_i} (a set of questions that have not been asked and are associated with c_i). We then define the final policy parametrized by θ: $\pi_\theta(a|B_{h_t}) = \frac{\mathbf{p_c}[i]}{|U_{c_i}|}$ for $c_i \in C$ and $a \in A$. Our policy $\pi_\theta(a|B_{h_t})$ allows us to select the next question to query and add the next question-response pair (x^t, y^t) to the response history. We then update B_{h_t} based on the updated response history using the LBP algorithm. We calculate the reward for the current time step $r_t = Acc(Binary(B_{h_{t+1}})) - Acc(Binary(B_{h_t}))$.

We use REINFORCE policy gradient method [25, 29] to learn our policy π_θ parametrized by θ. In each episode corresponding to a single student, the RL agent performs an update as follows. First, an initial state s_0 (the initial belief that each concept has a 0.5 probability of being known by the student) is generated. Then, the policy π_θ is executed until the episode ends, generating a sequence of experience given by $(s_t, a_t, r_t)_{t=1,2,...,T}$. Then, in this episode, for each $t \in \{1, 2, ..., T\}$, we use the following gradient update with η as learning rate:

$$\theta \leftarrow \theta + \eta \cdot \underbrace{\left(\sum_{\tau=t}^{T} r_\tau\right) \cdot \left(\nabla_\theta \log\left(\pi_\theta(a_t \mid s_t)\right)\right)}_{\text{gradient at time step } t \text{ in an episode}} \quad (5)$$

In experiments, we use the architecture used in [7]. Specifically, the policy network is a 3-layer fully connected neural network with the following architecture: the input layer has $n = 57$ units for B_h; the first and second hidden layers have 128 hidden units; and the output is a vector $\mathbf{p_c} \in \mathbb{R}^n$ where $n = 57$ to produce a probability of selecting each of the 57 concepts. The first two hidden layers use ReLU activations, and the final layer uses the softmax function to ensure probabilities sum to 1. We use ADAM [14] optimizer for training.

4. EXPERIMENTAL EVALUATION

We first evaluate and compare our quizzing policies (QPs) using a synthetic dataset. We then apply our QPs to the Eedi dataset from the NeurIPS 2020 Education Challenge.

4.1 Simulations

We simulate virtual students taking the assessment quiz and test how well we can predict students' knowledge states in a controlled setting using different QPs.

4.1.1 The synthetic dataset

We generate a dataset consisting of 24,000 simulated student knowledge states. To do so, we first construct a graph for representing the student knowledge state that we aims to infer (see Section 2.2) and then get a probability distribution over the binary variables in the knowledge state that satisfies a set of assumptions about the student's knowledge. We then sample ground-truth student knowledge state values from the probability distribution. In this simulation, we use the same 57 concepts in the Eedi dataset (described in Section 4.2.1) for constructing the graph. We assume some of these concepts have different levels of difficulty, and similar concepts are more likely to be assigned the same knowledge state values.[2] Based on these assumptions, we assign a value of difficulty to each of the 57 concepts. We define $\textit{difficulty}_{c_i}$ = 1− the average correctness of the concept c_i across all students' answers in the Eedi dataset.[3] We run the LBP algorithm on the constructed graph to get a probability distribution from which we sample student knowledge states. Specifically, the output of the LBP algorithm gives the probability of knowing each concept, and we sample values of 0 or 1 for each concept to generate the ground-truth student knowledge states in our synthetic dataset.

[2]Although our assumptions might not hold in a real-world setting, the goal of this experiment is to compare different QPs and investigate the potential of QP-RL for learning a strategy tailered to a pre-defined knowledge structure. For instance, compared to the heuristic approach QP-Uncertain, QP-RL should learn to select the questions that are not only uncertain but can also give more information about other questions that are not selected (e.g., selecting questions corresponding to concepts that are connected with a lot of the other concepts).

Table 1: Test performance of different QPs on the synthetic dataset. QP-Uncertain achieves a better performance than QP-Random, and QP-RL improves over QP-Uncertain significantly.

QP	Accuracy
QP-RL	0.721 ± 0.004
QP-Uncertain	0.700 ± 0.002
QP-Random	0.675 ± 0.003

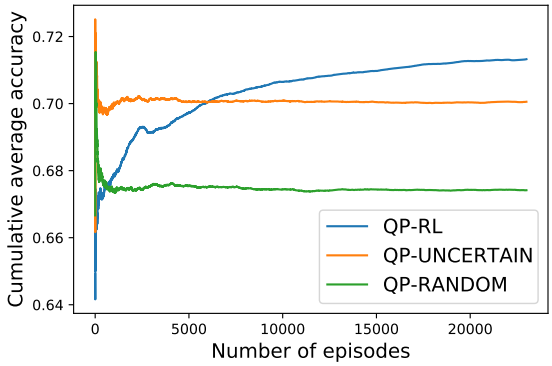


Figure 4: Training performance of QP-RL on the synthetic dataset compared to heuristics. QP-RL improves over QP-Random and QP-Uncertain after about 6,000 episodes of training. The cumulative average accuracy at each episode is calculated as the average accuracy across all previous episodes. It is important to note that QP-Random and QP-Uncertain are fixed policies that are not being trained. The cumulative average accuracy for the first few episodes might seem noisy due to small sample size.

4.1.2 Results

We split the dataset into 23,000 students as the training set and 1,000 students as the test set. We train QP-RL until the cumulative average accuracy converges. Figure 4 shows the training performance of QP-RL compared to fixed heuristics. After training, we run each QP 10 times on the test set to calculate the average accuracy and standard deviation across these 10 trials, see Table 1. Although QP-RL leads to a 2% gain in accuracy compared to QP-Uncertain, it requires a moderate amount of training data ($> 6,000$ students in this case). QP-Uncertain is a less optimal strategy but can achieve a reasonably good performance without any training data. These results provide initial evidence that QP-RL can learn an effective QP, and the performance can be improved further with more data.

[3]For simulations, one could also try other difficulty values, but it does not matter which specific difficulty value we assign to each concept because the goal is to model a setting where we have concepts of varying levels of difficulty.

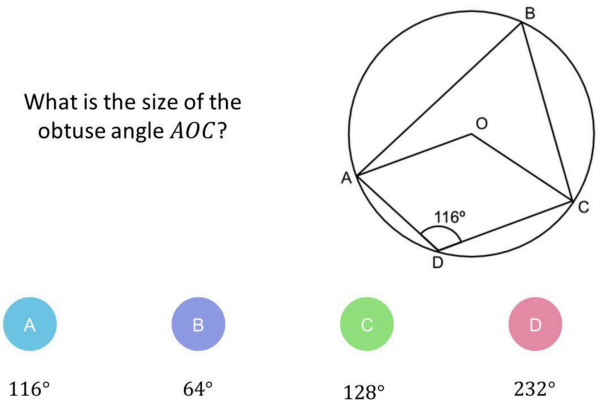


Figure 5: An example of a question in the Eedi dataset [27]. For each multiple-choice question, exactly one choice is correct.

4.2 NeurIPS 2020 Education Challenge

We then apply our QPs to one of the tasks in the NeurIPS 2020 Education Challenge (see Section 4.2), which is to obtain a limited set of answers from each student for predicting the correctness of their answers to the remaining questions. Our approach to this challenge is to first infer a student's knowledge state using Algorithm 1 and then predict the student's responses to the remaining questions based on the inferred knowledge state.

4.2.1 The Eedi dataset

The Eedi dataset contains student responses to multiple-choice questions (see Figure 5) on various math topics, which was collected between September 2018 and May 2020. It contains 948 questions and a total number of 1,508,917 responses to these questions from 6,148 students. The dataset is split into the training set (4918 students), the validation set (615 students), and the test set (615 students).

Each question in the dataset is associated with a list of subjects. Each subject covers an area of mathematics. These subjects are arranged in a tree structure by experts based on the generality of the subjects. For instance, "Fractions" is the parent subject of "Multiplying Fractions" and "Simplifying Fractions". For simplicity, we only consider the most granular subject (i.e., the leaves in the tree) as the concept that each question corresponds to. The 948 questions correspond to 57 unique concepts. We consider concepts that share the same super-concept (i.e., parent) to be similar (see Figure 1).

4.2.2 Student performance prediction

To predict a student's responses to unseen questions based on the inferred knowledge state, we propose a neural network-based model that takes in the belief about the student's knowledge B_{h_T} at time $T = 10$ (our belief about their knowledge after we have asked 10 questions) and outputs the probability of answering each of the 948 questions in the dataset correctly. The student performance prediction model is a 3-layer fully connected neural network with the following architecture: the input layer has $n = 57$ units for B_{h_T}; the first hidden layer has 256 hidden units; the second hidden layer has 512 units; and the output is a vector $\hat{\mathcal{Y}} \in \mathbb{R}^m$ where $m = 948$ to represent the probability of correctness for each of the 948 questions. The first two hidden layers use ReLU activations, and the final layer uses the sigmoid function to ensure the output values are between 0 and 1. We use ADAM [14] optimizer for training. We convert the output probabilities into binary values of 0 or 1 (0 if the probability is less than 0.5, 1 otherwise) and calculate the prediction accuracy using Equation 4. We train the model using randomly selected queries until the validation accuracy converges. The model parameters are updated based on binary cross-entropy loss.

Table 2: Test performance different QPs on the Eedi dataset. QP-RL improves slightly over QP-UNCERTAIN.

QP	Accuracy
QP-RL	0.690 ± 0.005
QP-UNCERTAIN	0.680 ± 0.003
QP-RANDOM	0.684 ± 0.003

4.2.3 Results

Given a trained performance prediction model from Section 4.2.2, we then train QP-RL using the difference in final prediction accuracy between time steps as reward signals: $r_t = Acc(Binary(\hat{\mathcal{Y}}_t)) - Acc(Binary(\hat{\mathcal{Y}}_{t-1}))$. After training, we run each QP 10 times on the test set to calculate the average accuracy and standard deviation across these 10 trials. Table 2 shows that QP-RL improves slightly over QP-UNCERTAIN, but the difference between QP-RL and QP-RANDOM is not significant. Results in Section 4.1.2 show that in a more controlled setting, QP-RL already requires a moderate amount of training data ($> 6,000$ students) to improve over heuristics. However, we only have training data from about 5,000 students in this experiment. Learning a QP from real students' data that are noisy is more challenging, and it may be the case that improving QP-RL further would require a much larger dataset. Even though QP-RL seems to require a substantial amount of training data, this is a one-time training, and the learned policy can be applied to future students.

5. CONCLUSION

Student assessment is a crucial component of many online education systems for improving student learning outcomes. Inferring student knowledge state by quizzing poses a technical challenge: maximizing accuracy while minimizing the quizzing cost. In this paper, we show initial evidence that reinforcement learning (RL) provides a potential solution, improving over heuristics given sufficient training data.

There are several research directions for future work. Further gains in accuracy could be achieved by exploring more powerful RL techniques and more complex student knowledge modeling techniques. In this work, we model all concepts that share the same super-concept as having the same relationship; however, there could be prerequisites as well as weaker and stronger relationships in reality. It would be important to study whether varying the influence values between concepts would lead to gains in model performance.

6. REFERENCES

[1] U. Z. Ahmed, M. Christakis, A. Efremov, N. Fernandez, A. Ghosh, A. Roychoudhury, and A. Singla. Synthesizing tasks for block-based programming. In *NeurIPS*, 2020.

[2] J. Bassen, B. Balaji, M. Schaarschmidt, C. Thille, J. Painter, D. Zimmaro, A. Games, E. Fast, and J. C. Mitchell. Reinforcement learning for the adaptive scheduling of educational activities. In *CHI*, pages 1–12, 2020.

[3] M. Chi, K. VanLehn, D. Litman, and P. Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1):137–180, 2011.

[4] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[5] S. Doroudi, V. Aleven, and E. Brunskill. Where's the reward? *International Journal of Artificial Intelligence in Education*, 29(4):568–620, 2019.

[6] Eedi. Eedi: Online maths lessons with live teacher support. `https://eedi.com`.

[7] A. Efremov, A. Ghosh, and A. Singla. Zero-shot learning of hint policy via reinforcement learning and program synthesis. In *EDM*, 2020.

[8] J.-C. Falmagne, M. Koppen, M. Villano, J.-P. Doignon, and L. Johannesen. Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review*, 97(2):201, 1990.

[9] M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach. *CoRR*, abs/1708.02383, 2017.

[10] R. Gilad-Bachrach, A. Navot, and N. Tishby. Query by committee made real. In *NIPS*, volume 5, pages 443–450, 2005.

[11] M. R. Gormley and J. Eisner. Structured belief propagation for nlp. In *ACL*, pages 5–6, 2015.

[12] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, pages 417–424, 2006.

[13] A. Iglesias, P. Martínez, R. Aler, and F. Fernández. Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. *Knowledge-Based Systems*, 22(4):266–270, 2009.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[15] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR*, pages 3–12, 1994.

[16] Y. Liu. Active learning with support vector machine applied to gene expression data for cancer classification. *Journal of chemical information and computer sciences*, 44(6):1936–1941, 2004.

[17] T. Mandel, Y.-E. Liu, S. Levine, E. Brunskill, and Z. Popovic. Offline policy evaluation across representations with applications to educational games. In *AAMAS*, pages 1077–1084, 2014.

[18] R. Moskovitch, Y. Elovici, and L. Rokach. Detection of unknown computer worms based on behavioral classification of the host. *Computational Statistics & Data Analysis*, 52(9):4544–4566, 2008.

[19] K. Pang, M. Dong, Y. Wu, and T. Hospedales. Meta-learning transferable active learning policies by deep reinforcement learning. *CoRR*, abs/1806.04798, 2018.

[20] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[21] J. Rollinson and E. Brunskill. From predictive models to instructional policies. In *EDM*, 2015.

[22] N. Roy and A. McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML*, pages 441–448, 2001.

[23] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *COLT*, pages 287–294, 1992.

[24] A. Singla, S. Tschiatschek, and A. Krause. Actively learning hemimetrics with applications to eliciting user preferences. In *ICML*, pages 412–420, 2016.

[25] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[26] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[27] Z. Wang, A. Lamb, E. Saveliev, P. Cameron, Y. Zaykov, J. M. Hernández-Lobato, R. E. Turner, R. G. Baraniuk, C. Barton, S. P. Jones, et al. Diagnostic questions: The neurips 2020 education challenge. *CoRR*, abs/2007.12061, 2020.

[28] J. Whitehill and J. Movellan. Approximately optimal teaching of approximately optimal learners. *IEEE Transactions on Learning Technologies*, 11(2):152–164, 2017.

[29] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[30] M. Woodward and C. Finn. Active one-shot learning. *CoRR*, abs/1702.06559, 2017.

[31] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127, 2015.

[32] J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end autonomous driving. *CoRR*, abs/1605.06450, 2016.

From Detail to Context: Modeling Distributed Practice Intensity and Timing by Multiresolution Signal Analysis

Cheng-Yu Chung
Computer Science
Arizona State University
Cheng.Yu.Chung@asu.edu

I-Han Hsiao
Computer Science
Arizona State University
Sharon.Hsiao@asu.edu

ABSTRACT

The distributed practice effect suggests that students retain learning content better when they pace their practice over time. The key factors are practice dosage (intensity) and timing (when to practice and how in between). Inspired by the thriving development of image recognition, this study adopts one of the successful techniques, multiresolution analysis (MRA), to model distributed and spaced practice (SP). We consider a sequence of practice sessions as a signal of the student's learning strategy. Then, we apply the stationary wavelet transform (SWT) to extract practice patterns spaced by three periods: small, medium, large. The result reveals a positive correlation between the small-spaced practice and the exam grade. The benchmark against baseline feature models shows that the SP patterns significantly improve the goodness-of-fit and complements the baseline models. This work successfully demonstrates 1) the use of MRA in modeling sequential patterns by event intensity and event timing; 2) the MRA approach can be used as an alternative method to improve existing student models of practice effort.

Keywords

distributed practice effect, testing effect, stationary wavelet transforms, signal multiresolution analysis, feature extraction

1. INTRODUCTION

In the midst of blended and distance learning environments, it is increasingly important for students to manage their time efficiently. Numerous researchers have proposed and developed various student models to capture how students utilize their time during the learning process. The results have shown that distributed practice is a simple but effective time-management strategy for learning [5]. Essentially, distributed practice comprises the testing and spacing effects, which suggest that the retention of information increases when the learner practices retrieving it in multiple spaced-out practice sessions [3].

Optimal distributed practice requires a combination of both the *intensity* and the *timing* of the practice events. In other words, an expressive student model must capture the intensity of practice sessions spaced by different periods. Although the two features appear to be straightforward, it is not easy to incorporate them in a sequential behavior model. For example, typical sequence analysis or sequential pattern mining would expect discrete input data and extract common patterns in the data according to the sequence support (the number of occurrences). Finding a meaningful and interpretable threshold is usually an ad-hoc process and particularly challenging [4]. A great threshold value may increase the chance of losing detail, and a small value may introduce more noises and miss the context. In the case of distributed practice, when the practice sessions are far apart, such a frequency-based approach will require more data to ensure sufficient within- and between-sequences support for a pattern of interest. To address this modeling challenge, we are motivated to explore an alternative computational method to capture the detail as well as the context, which can capture both the intensity and the timing of events at the same time.

We rationalize that a student's practice sessions distributed over a timeline resemble a signal to her/his learning process where the strength of learning is quantified as the increasing or decreasing values about the occurrences of the underlying events. With this definition, we can utilize a signal processing tool to extract the structural variation which approximates distributed practice patterns. In this work, we adopt the stationary wavelet transform (SWT) algorithm for this purpose. SWT is a widely-used signal processing tool in an application such as image pattern recognition. The algorithm decomposes an input signal into multiple components and represents the original signal by information at different resolutions. With the emphasis on the structure, we believe that SWT will allow us to overcome the challenge where the amount of sequential data may not be big enough to maintain the sequence support. Additionally, applying SWT as a feature extraction method also allows us to examine structural nuances in behavior sequences.

2. RELATED WORK

2.1 Sequence Analysis in Educational Data Mining

Cheng-Yu Chung and I-Han Hsiao "From Detail to Context: Modeling Distributed Practice Intensity and Timing by Multiresolution Signal Analysis". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 540-546. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

A *behavior sequence* is a chronicle of an activity. It describes a collection of events, and the order of them is meaningful. We can choose different features to characterize such a sequence, e.g., types of events, arrangements of events, time gaps between events. The features directly affect what we can find out from the analysis. Sequence analysis, in general, can refer to any data model that involves a kind of behavior sequences its characterization. Extensive research in EDM has been using behavior sequence analysis to model students' development of knowledge or skills.

The most intuitive approach is sequential pattern mining, which aims to discover repeated string patterns, alignments, or the very next possible items [11]. For example, Gitinabard et al. characterize behavior sequences by students' interactions with online tools [7]. They map the interaction sequences to study habits and use sequence patterns to differentiate the high-performing and low-performing students. Dermy and Brun argue that the time interval is the key to model students' activities [4]. They characterize behavior sequences by time intervals between events and formalize the temporal information in sequential pattern mining. Their experiment suggests a strong correlation between the students' activities and the time information.

One research gap we notice is that most of the reviewed works focus on the behavior sequences at a single time scale. For example, for a given behavior sequence $e_1, e_2, ..., e_t$ where e_i is an event that occurs at time i. A typical sequence analysis focuses on the relationship of adjacent events e_{j-1}, e_j, e_{j+1} where $j \in 1, ..., t$. Since the step size is 1, sometimes such a sequence is called 1-sequence. Following this setting, a pattern must be a consecutive 1-sequences that meets predefined criteria, e.g., the support. One limitation of 1-sequences is that they cannot capture an inconsecutive event. Such an inconsecutive event can provide a coarser view of the behavior sequence, therefore the context. Indeed, we can try to increase the step size to have 2-sequences, 3-sequences, or k-sequences where $k \in \mathbb{Z}$. Nonetheless, the increment of step size inevitably reduces the number of k-sequences we can find in a dataset. This situation may exclude potential sequences of interest due to the threshold of the support or the shortage of data. To tackle this challenge, we investigate an alternative model that focuses on the structural information of behavior sequences.

3. MULTIRESOLUTION SIGNAL ANALYSIS

In pattern recognition, the information of a given object usually is determined by the variations of signal intensity. For example, we can recognize a building as a building in an image because the distinct contours and shapes are formulated by their unique signal value sequences and different from the other objects. Such signal *features* are essentially sequences of values (sets of numbers) where a variation of intensity could suggest a potential event of interest, e.g., a change of shapes or colors. However, because the objects to analyze may have different shapes and sizes, the feature extraction must consider "how far away" an event is from its neighborhood to recognize the objects' structures at multiple resolutions. The field of computer vision and signal processing have developed various methods to address this challenge. One of which is the multiresolution analysis and wavelet transforms, which fit in the scope of this research.

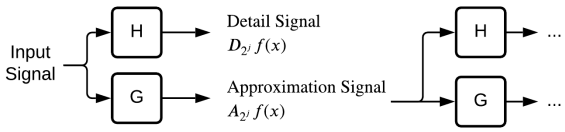


Figure 1: The Decomposition of Multiresolution Analysis. The process consists of two filters: the high-pass filter H and the low-pass filter G. They iteratively extract the detail signal and the approximation signal at the resolution 2^j from the input signal $f(x)$ until a maximum level L. We can associate the interpretation of the detail signal to the underlying time scale. For example, say the sampling rate of the input signal is 1. The detail signal at level 1 (a coarser level) denotes the information from the frequency band $[1/2, 1/4]$.

The multiresolution analysis (MRA) is a hierarchical framework that describes how to decompose a signal from fine to coarse levels [12]. The decomposition consists of a high-pass filter (H) and a low-pass filter (G). They are a pair of quadrature mirror filters and have the following relationship: $g(n) = (-1)^{1-n} h(1-n)$ [12]. The high-pass filter extracts impulses, and meanwhile, the low-pass one retains the other information. This process is also known as Discrete Wavelet Transform (DWT). By convolution ($*$), the filtering process iteratively produces series of detail signals ($D_{2^j} f(x)$) and approximation signals ($A_{2^j} f(x)$) for the input signal $f(x)$:

$$D_{2^j} f = (f(u) * \psi_{2^j}(-u))(2^{-j} n) \tag{1}$$

$$A_{2^j} f = (f(u) * \phi_{2^j}(-u))(2^{-j} n) \tag{2}$$

where $n \in \mathbb{Z}$. The high-pass and low-pass filters rely on a wavelet function (ψ) and a scaling function (ϕ) that translate and scale the input signal at different resolutions, respectively. We illustrate the whole filtering process in Figure 1 for reference. See [2] and [12] for more details about the math properties of the wavelet function and the scaling function.

3.1 Analyzing Distributed Practice via Signals

In this study, we focus on the practical implication of MRA and illustrate how it can help identify students' distributed practice patterns. Students, especially those in an online learning or a blended learning environment, usually have greater flexibility in self-pacing their studies. In other words, they can watch the lecture videos and practice quiz questions anytime at their convenience. This nature makes it challenging to analyze their behaviors on the timeline.

For example, in scenario A, when a semester is two to three months long, we may find out that the students' practice sessions are sparse and do not follow one unified schedule. This makes the time of sessions less discriminating in finding common behavioral patterns. Thus, the researcher may choose to ignore the time feature. An alternative approach (scenario B) is aggregating the practice sessions by a priori

assumption (e.g., students always study on a week-by-week basis or right before a deadline). However, all the above approaches may inevitably lose some detail about how exactly the students utilize their schedules, either missing discriminating patterns over time (scenario A) or limited to those strictly abiding by the class-paced schedule (scenario B).

To model students' distributed practice behavior over time, his/her behavior can be denoted as a sequence of the events, f with T discrete time steps: $f = \{e_1, e_2, ..., e_T\}$ where e_i for $1 \leq i \leq T$ can be any activity event of our interest. A student distributes his/her practice sessions at different rates or frequencies according to his/her preferences, path, or pace. This representation is like a signal and enables the feasibility to apply a signal processing algorithm.

Similar to the pattern recognition in computer vision, a student's practice sessions are like the shapes and colors that may evolve according to the sequences of signals. The sessions may have different sizes, i.e., time gaps between any two sessions. In other words, we aim to extract *distributed spaced practice* (SP) that are subsets of the input behavior sequence: $SP_k \subseteq f$ where $k \in \mathbb{N}$ and any two consecutive event items $\{e_i, e_j\} \subset SP_k$ are spaced by k time steps. Following this idea, MRA is used to extract such a "feature" from sequences of practice sessions, and thereby interpret the output as distributed practice patterns. For a practice signal at sampling rate = 1/day, the output signals can represent the information at coarser rates, e.g., 1 per 4 days and 1 per 8 days.

3.2 Stationary Wavelet Transform

The output of DWT are signals that represent information at different resolutions (or frequencies). The typical implementation of DWT keeps downsampling the input signal to obtain the detail signal and the approximation signal at each resolution [12]. Therefore, the transform is time-variant. The detail signal at one level is a half shorter than the one at the previous level. This property may cause a misalignment in time/frequency, which will make the decomposition generate fewer feature values for analysis. In this study, we follow an alternative implementation of MRA, Stationary Wavelet Transform (SWT), which is time-invariant. SWT replaces the downsampling by upsampling at each step [6]. Research has shown that SWT can improve the approximation and a preferred approach for applications like breakdown point detection and denoising [1].

4. DATASETS

To evaluate the method, we use two semesters' datasets from the same undergraduate class offered in a four-year university in the United States: Spring 2018 (SP18) and Fall 2018 (FA18). Both sessions lasted about 3-month. The class was a typical lecture-style in-person class with weekly assignments and monthly exams. The two sessions were practically identical, having exactly the same syllabus, same instructor, same teaching assistants, except for minor adjustments to the exam questions. Note most students in FA18 shared a similar background in engineering because the class was a required class for first-year engineering students. In SP18, there were more students from non-engineering schools, which resulted in much more diverse student background.

An online practice platform was introduced to the students at the beginning and available throughout the semester. On the platform, students could take multiple-choice questions to practice and review the class content. For any given practice question, the students had unlimited chances to retry; for any attempt, the corrective feedback (correct answer) would be provided upon submission. The questions served like so-called "tasks" in the context of tutoring systems [16]. Each of the tasks aims to help the student master some knowledge (or embedded knowledge components). However, the practice activity is different from working with assignments: there is no "hard deadline" by which the students must complete the practice questions. The students can practice on the platform as a kind of self-assessment [13]. In other words, the activity is "self-paced" [18] and aligned with the actions of reviewing slides, taking quizzes, or other practices that students can do for their benefit whenever they want.

The students' practice activities were logged as transactions of events, including the timestamps, the questions, and the correctness of the attempts. We processed and transformed the data into sequences of daily practice intensity. Here, the term "intensity" refers to the number of unique questions solved by a student. Each day is assumed to be a complete practice session. The sequence of daily intensity thereby resembles a discrete-time signal sampled at a constant rate equal to 1 sample per day. We excluded some students' data from the analysis due to low usage (those who only had only one practice session throughout the semester). An overview of the datasets is described in Table 1.

In this study, the exam letter grade is used as the students' learning performance index. The exam letter grade ranges from A ($M \geq 90$), B ($80 \leq M < 90$), to C/D/F ($M < 80$) where M is the raw average of three exam scores.

5. REPRESENTING DISTRIBUTED PRACTICE BY SWT SIGNALS

There are several parameters required for our model pipeline: the wavelet for SWT, the padding scheme, the maximum decomposition level, and the penalty of change point detection. The Haar wavelet is adopted in the SWT algorithm implementation, due to the simplest form of wavelet [14]. It creates a shape like a step function that produces 1, 0, and -1, following the formula

$$\psi(x) = \begin{cases} 1 & \text{if } 0 \leq x < \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This property makes it a good option for detecting edges (e.g., sudden signal transitions or changes) [17] in discrete signals like the datasets in this study. The implementation of SWT used in this study requires the length of input to be a multiple of 2^L where L is the maximum number of levels to decompose [9]. To meet this requirement, we preprocessed all input sequences by adding a prefix of zeros. In our experiment, we found that the SWT signals at $L > 3$ did not work. It was likely due to short input sequences. Therefore,

Dataset	# of Students	# of Included	Max Length of Sequence (Days)	# of Questions	M (SD) of Intensity
SP18	121	76 (63%)	93	96	0.26 (0.36)
FA18	200	67 (34%)	82	95	0.32 (0.49)

Table 1: Statistics of the Two Datasets

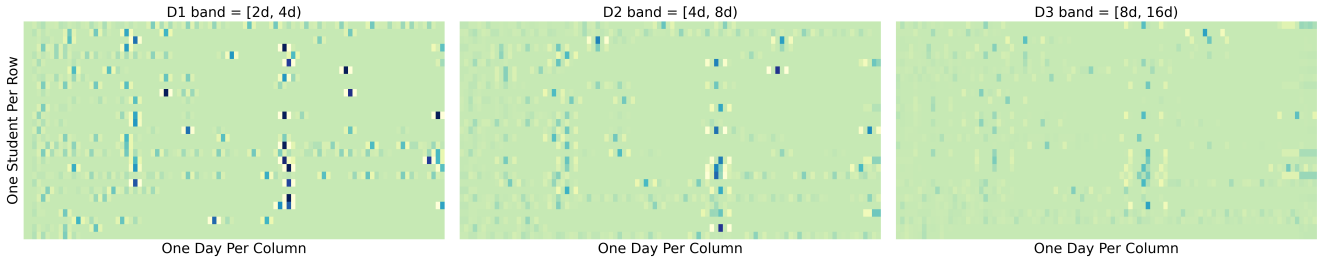


Figure 2: The SWT Signals in the SP18 Dataset. From left to right, the SWT signals D1, D2, D3 capture practice sessions in the period bands [2d, 4d], [4d, 8d], and [8d, 16], small-, medium- and largely-spaced respectively. These signals capture practice sessions spaced by different periods. From D1 (more detail) to D3 (more context), we can see the focus gradually spreads out when the level increases. For readability, the plot excludes practice sequences not having any change points in the SWT signals.

we set L = 3 in our experiment. Once the SWT algorithm is built, we applied the change point detection algorithm with the penalty = 0.5 to search for sudden changes in the SWT signals [8]. We decided on this penalty value by maximizing the group difference (Section 5.2) and the goodness-of-fit of regression (Section 6.2). The experiment program and data are available at the link for future work[1].

5.1 Characteristics of SWT Signals

The SWT algorithm decomposes the input signal by multi-level filtering. Filtering at a level k extracts information at the frequency band $[1/2^k f, 1/2^{k+1} f]$ where f is the sampling rate of the input. In our datasets, because the sampling rate is 1 sample per day (1 cycle per day), the three decomposition levels (Dk where k = 1,2,3) filter the input in the frequency bands $[1/2, 1/4]$, $[1/4, 1/8]$, and $[1/8, 1/16]$. In other words, the algorithm filters the input into the period (the duration of time of one cycle) bands D1=[2(d)ays, 4d], D2=[4d, 8d], and D3=[8d, 16d]. We map these three bands to small-spaced, medium-spaced, and largely-spaced practice patterns, respectively. Following this interpretation, we expect the SWT signals to identify students' practice sessions spaced by different periods. For example, D1 can identify sessions spaced by 2 to 4 days, which are small-spaced practice.

To further illustrate this characteristic, Figure 2 demonstrates what the algorithm found in the SP18 dataset. The visualization shows the SWT signals at the three levels. We can see that D1 highlights small-spaced practice sessions. The D2 and D3 signals spread their focus and "blur" the sequences not fitting their period bands. Note, there may be redundancy in the information captured by different components. For example, an input sequence having meaningful change points in D3 can also have ones in D1. Overall, the information about practice sessions at different levels provides an insight into how the students distribute their practice over time. In our analysis of distributed practice patterns, we use the number of change points as the *feature* to represent the information from the three SWT signals.

For readability, we use the lower bound of the frequency band to denote the spaced practice patterns. We call the practice patterns found in the D1, D2, D3 signals *2SP* (2-day spaced practice), *4SP*, and *8SP* patterns, respectively. For reference, the input daily practice sessions are called 1SP. In the SP18 dataset, the means (M) / standard deviation (SD) from the three levels are 2SP = 1.83/2.68, 4SP = 1.79/2.63, and 8SP = 1.75/2.63. In the FA18 dataset, the values are 2SP = 1.12/2.29, 4SP =1.27/2.14, and 8SP = 1.37/2.30.

5.2 Marginal Relationship of Spaced Patterns with Exam Grades

We analyzed the relationships between the practice patterns and student grades by the marginal distribution. The Kruskal-Wallies H-test was applied to test if the groups had the same population median (Figure 3). The method was selected because the sample size was small, and therefore the sample might not follow the normal distribution. The results showed that there was only 2SP that appeared to be significant for both datasets (SP18: H=8.89, p=0.01; FA18: H=7.95, p=0.02). The visualization of the distribution showed that in SP18 A students had a higher 2SP (M = 3.12, SD = 3.11) than C (M = 1.50, SD = 2.32) and B (M = 0.72, SD = 1.79); in FA18, the B students had a higher value (M = 2.17, SD = 3.05) than A (M = 0.62, SD = 1.50) and C (M = 0.41, SD = 1.14).

There are more spaced patterns discovered for B students in FA18 but not A students, which suggests there could be other factors in the correlation of their practice with even higher exam grades. For example, engineering and non-engineering students may have/need different practice strategies adapted to their learning conditions. Despite this slight difference across the two semesters, if we focus on the difference between the higher-performing students (A/B) and the C/D/F ones, the result consistently suggests a positive correlation between exam grades and small-spaced prac-

[1]https://github.com/rickchung/edm21-msa

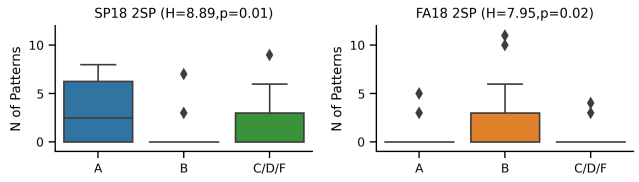


Figure 3: Marginal Distribution of the SP patterns from the Three Grade Groups. The Kruskal-Wallies H-test found that only 2SP was significant in both datasets. The result consistently showed that higher-performing students (A in SP18 or B in FA18) had more small-spaced practice than the C/D/F ones in SP18 and FA18.

tice.

5.3 Quantifying the Schedule of Distributed Practice

We have seen how the SP patterns can help identify practice spaced by different periods. However, this feature alone does not depict the entire picture of distributed practice strategies. Another key factor in the distributed practice effect is the timing of practice. We develop an index to quantify the skewness in practice schedules and investigate its correlation to exam grades. One simple measure of the skewness is the *lag time*. We can use the *lag time* between the occasion of a practice session and a specific event of interest (e.g., exam dates, assignment deadlines) to model the schedule skewness. Due to programming is inherently accumulative, a later exam covers the content from all the previous exams, we cannot assert that a practice session only affects the upcoming exam. Considering this case, we focus on the time lag since the beginning of the semester. Specifically, for an SWT signal at level i, D_i, we can compute the lag of days between the start of the semester and the occurrences of change points. Then, we can transform a practice sequence into a sequence of lags $\{T_1^{D_i}, T_2^{D_i}, T_3^{D_i}, ..., T_n^{D_i}\}$. To know where on the timeline the student has more practice, we compute the sample mean, $\mu_T^{D_i}$. The number, therefore, represents how far the schedule is away from the beginning of the semester. We further divide the number by the total number of days (N_{day}) in the semester for interpretation. The equation of the schedule skewness is defined as

$$SS = \frac{\mu_T^{D_i}}{N_{days}} \tag{4}$$

When a student has all his/her practice sessions early in the semester, SS will be close to zero. If s/he has more practice sessions over the middle of the semester, SS will be some value over 0.5. We can apply the formula to the input signal (1SS) and the SWT signals (2SS, 4SS, 8SS). The result will indicate the schedule of different spaced-practice patterns.

6. MIXED PRACTICE EFFECTS IN MULTIVARIATE ANALYSIS

A distributed practice strategy is multifaceted. The univariate analysis is insufficient because it does not consider the confounding variables. There are two cases remain unclear.

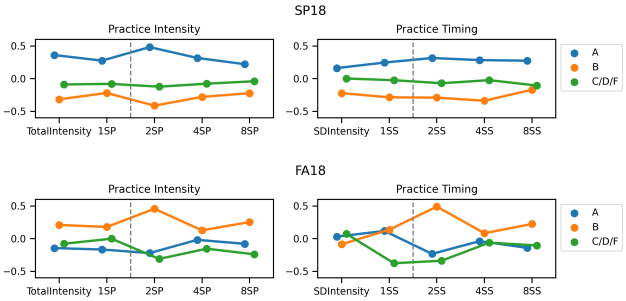


Figure 4: The Standardized Values (Means) and the Interactions of the Basic and Experiment Features. The plot groups the features into intensity and timing according to their functions. The y-axis shows the standardized values (by $(X - M_X)/SD_X$ in each feature category X) for between-features comparison. The vertical dashed lines separate the basic and experiment features. The A students in SP18 have the highest TotalIntensity. On the contrary, the B students in FA18 have the highest TotalIntensity. Although the basic features somehow correlate with the experiment features, we can find more discriminating differences in the experiment spaced-practice patterns.

First, the C/D/F students from SP18 do not have better performance, even though they put efforts into practice just like the better performing students do (A students). Second, in FA18, the analysis does not explain the practice strategy of the A students. They achieve a good grade but do not show significantly more SP. These cases suggest that there could be other factors in the distributed practice effects.

Following this idea, we try to use multivariate analysis that includes experiment SWT features and commonly-used basic features. The basic set comprises the following. For the practice intensity, we use the total number of questions solved (TotalIntensity) and the total number of daily practice sessions (1SP). For the practice schedule, we use the standard deviation of the daily intensity (SDIntensity) and the SS of daily practice sessions (1SS). For reference, we plot the standardized values of the features in Figure 4. The figure shows that although the basic features correlate with the experiment features, we can potentially find more discriminating difference in the spaced practice patterns between the grade groups.

6.1 Assessing the Marginal Effects by Multinominal Logit Regression

To understand the relationship between multiple feature and exam grades, we use the multinominal logistic regression and investigate the marginal effects of the feature values. The *multinominal logistic regression* (MLogit) is a generalized version of the logistic regression for multiclass classification problems [15]. We can use MLogit when the dependent variable in a query is nominal (categorical) and has more than two possible categories. The setup of MLogit is similar to the logistic regression. We assume a linear relationship between the independent variables (predictors), X, and the dependent variable (response), Y, and model the probability of the $Y \in \{y_1, ..., y_k\}$ by the logistic function (*sigmoid*)

and k-1 sets of weights (w_k for the label y_k):

$$P(Y = y_k|X = X_1, ..., X_n) = \frac{exp(w_{k0} + \sum_i w_{ki}X_i)}{1 + \sum_j exp(w_{j0} + \sum_i w_{ji}X_i)} \quad (5)$$

We can obtain the prediction by picking up the class with the highest probability. The main advantage of MLogit over other classification techniques is the interpretability. We can explain the contribution of individual features to the output probability (dx/dy) similar to the linear regression [15]. In the analysis, we set Y as the grade groups (A, B, C/D/E) and examine the marginal effects with respect to X when the model fits different sets of predictors.

6.2 Comparing Alternative Models

To understand the capability and limitation of the SWT model of distributed practice, we use MLogit to fit various baseline and experiment feature sets. Afterward, we benchmark the quality of these models by the goodness-of-fit. Due to MLogit does not use the standard R^2, we use the measure of the goodness-of-fit by *McFadden's pseudo* R^2 [10]. McFadden's pseudo R^2 uses the formula

$$R^2 = 1 - \frac{\ln L(M_{full})}{\ln L(M_{intercept})} \quad (6)$$

where L is the estimated likelihood. A small ratio of the two log-likelihoods (or a large McFadden's pseudo R^2) suggests that the full model is better than the intercept model. We can use this measure to benchmark one model against another if they fit the same data.

We compared the experiment and alternative baseline models. The result showed that none of the baseline models were competitive with even the simplest SWT model (using only 2SP, 4SP, 8SP). The best baseline model ($M_{BaseAll}$) used all the baseline variables and achieved $R^2 = 0.04$ in SP18 and $R^2 = 0.07$ in FA18. The simplest SWT model ($M_{ExpDose}$) achieved $R^2 = 0.07$ in SP18 and $R^2 = 0.09$ in FA18. The best experiment model (M_{ExpAll}) used all the SWT variables and achieved $R^2 = 0.12$ in both SP18 and FA18. Using all the baseline and experiment variables, the ensemble model ($M_{Ensemble}$) unsurprisingly outperformed all the other models and achieved $R^2 = 0.13$ and $R^2 = 0.23$ in SP18 and FA18, respectively.

6.3 Marginal Effects in the Regression Models

In SP18, $M_{ExpDose}$ found 2SP was a significantly-positive predictor for the A students (dx/dy = 0.07, p = 0.01). M_{ExpAll} also found that 2SP was a significant predictor for the A students (dx/dy = 0.10, p = 0.00). Besides, it found 4SS and 8SS were significant for the C/D/F students (dx/dy = 2.55, p = 0.02; dx/dy = -2.58, p = 0.03). In FA18, $M_{ExpDose}$ found 2SP was significantly-positive predictor for the B students (dx/dy = 0.12, p = 0.00). M_{ExpAll}, however, did not find any significant predictor.

Part of the result is similar to the analysis of marginal distribution. In SP18, an increase of small-spaced practice adds to the likelihood of A. In FA18, the same effect works for B. It is worth noting an additional finding in $M_{Ensemble}$ from SP18. When we control the intensity and SS, the model shows two extra significant predictors for the grade C/D/F: 4SS and 8SS. The marginal effect suggests that an increase/decrease in 4SS/8SS adds to/reduces the likelihood of C. Since an increase in SS means the schedule becomes later in the semester, these two findings somewhat suggest the same thing: students who practice early and space the practice largely are less likely to obtain C/D/F.

It is also worth noting that the one in FA18 improves the most from the best experiment model and reaches R^2 = 0.23. When predicting the A students, the model shows 1SS (dy/dx = 1.01, p = 0.00) and the total intensity (dy/dx = -0.02, p = 0.04) are significant predictors; when the model predicts the C students, 1SS is the only significantly-negative predictor (dy/dx = -0.90, p = 0.01). We do not find the same effect in any of the baseline models. The result complements a missing part of our analysis about the A and C students' practice strategies in FA18. It suggests that an increase in 1SS adds to the likelihood of A. Conversely, the same increase reduces the one of C/D/F. In other words, more early or late practices in the semester may reduce or improve the probability of C/D/F or A, respectively.

7. CONCLUSIONS

Students' practice behavior is challenging to model because they can practice anytime and do not necessarily follow a unified schedule. This study aims to build such a feature model that can help researchers describe the distributed practice behavior. We adopted the method from multiresolution analysis to extract patterns of our distributed practices, focusing on two factors in the distributed practice effect: intensity and timing. In the experiment, we applied the MRA model and extracted features that could represent practices spaced by different periods, including small (2-4 days), medium (4-8 days), and large (8-16 days). These three kinds of practice patterns were analyzed to explain their correlation to the exam grades. We found that students who practiced early and spaced the practice by the small and large periods were more likely to get a higher grade than C/D/F. Also, the students having more small-spaced practices throughout the semester (i.e., practicing more persistently) were more likely to get better exam grades. Additionally, the MRA model was benchmarked against baseline models. The result showed that the MRA model not only achieved a better goodness-of-fit than the baselines when working alone, but it could complement a baseline model and achieve better performance.

8. REFERENCES

[1] R. R. Coifman and D. L. Donoho. Translation-Invariant De-Noising. *Wavelets and Statistics*, pages 125–150, 1995.

[2] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1 1992.

[3] P. F. Delaney, P. P. Verkoeijen, and A. Spirgel. Spacing and Testing Effects. In *Psychology of Learning and Motivation - Advances in Research and Theory*, volume 53, pages 63–147. Elsevier Inc., 1 edition, 2010.

[4] O. Dermy, A. Brun, and U. D. Lorraine. Can we Take Advantage of Time-Interval Pattern Mining to Model Students Activity ? In *Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020*, pages 69–80, 2020.

[5] J. Dunlosky, K. A. Rawson, E. J. Marsh, M. J. Nathan, and D. T. Willingham. Improving Students' Learning With Effective Learning Techniques. *Psychological Science in the Public Interest*, 14(1):4–58, 1 2013.

[6] J. Fowler. The redundant discrete wavelet transform and additive noise. *IEEE Signal Processing Letters*, 12(9):629–632, 9 2005.

[7] N. Gitinabard, T. Barnes, S. Heckman, and C. F. Lynch. What will you do next? A sequence analysis on the student transitions between online platforms in blended courses. In *Proceedings of the 12th International Conference on Educational Data Mining, EDM 2019*, pages 59–68, 2019.

[8] R. Killick, P. Fearnhead, and I. A. Eckley. Optimal Detection of Changepoints With a Linear Computational Cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 12 2012.

[9] G. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, and A. O'Leary. PyWavelets: A Python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237, 4 2019.

[10] J. S. Long and J. Freese. *Regression models for categorical dependent variables using Stata*, volume 7. Stata press, 2006.

[11] N. R. Mabroukeh and C. I. Ezeife. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys*, 43(1):1–41, 11 2010.

[12] S. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 7 1989.

[13] E. Panadero. A review of self-regulated learning: Six models and four directions for research. *Frontiers in Psychology*, 8(APR):1–28, 2017.

[14] D. B. Percival and A. T. Walden. *Wavelet Methods for Time Series Analysis*. Cambridge University Press, Cambridge, 2000.

[15] R. L. Strawderman, A. C. Cameron, and P. K. Trivedi. Regression Analysis of Count Data. *Journal of the American Statistical Association*, 94(447):984, 9 1999.

[16] K. VanLehn. The Behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3):227–265, 2006.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–511–I–518. IEEE Comput. Soc, 2001.

[18] Y. Zhang, Y. Dang, and B. Amer. A Large-Scale Blended and Flipped Class: Class Design and Investigation of Factors Influencing Students' Intention to Learn. *IEEE Transactions on Education*, 59(4):263–273, 11 2016.

A Novel Algorithm for Aggregating Crowdsourced Opinions

Ethan Prihar
Worcester Polytechnic Institute
100 Institute Road
Worcester, Massachusetts
ebprihar@wpi.edu

Neil Heffernan
Worcester Polytechnic Institute
100 Institute Road
Worcester, Massachusetts
nth@wpi.edu

ABSTRACT

Similar content has tremendous utility in classroom and online learning environments. For example, similar content can be used to combat cheating, track students' learning over time, and model students' latent knowledge. These different use cases for similar content all rely on different notions of similarity, which make it difficult to determine contents' similarities. Crowdsourcing is an effective way to identify similar content in a variety of situations by providing workers with guidelines on how to identify similar content for a particular use case. However, crowdsourced opinions are rarely homogeneous and therefore must be aggregated into what is most likely the truth. This work presents the Dynamically Weighted Majority Vote method. A novel algorithm that combines aggregating workers' crowdsourced opinions with estimating the reliability of each worker. This method was compared to the traditional majority vote method in both a simulation study and an empirical study, in which opinions on seventh grade mathematics problems' similarity were crowdsourced from middle school math teachers and college students. In both the simulation and the empirical study the Dynamically Weighted Majority Vote method outperformed the traditional majority vote method, suggesting that this method should be used instead of majority vote in future crowdsourcing endeavors.

Keywords

Crowdsourcing, Similarity, Community Detection, Hierarchical Clustering

1. INTRODUCTION

Within online learning platforms and intelligent tutoring systems there is a tremendous opportunity to utilize knowledge of content similarity. Similar problems can help prevent cheating during exams by randomly selecting from multiple similar problems when students receive the exam, measure students' learning gains by spreading out similar problems between assignments, and measure the effects of instructional interventions by comparing a student's scores on similar problems before and after the intervention. Similar instructional material can be used to offer students choices in which instructional material they receive, which has been shown to increase engagement and achievement [7]. While it is possible to implement these methods with general knowledge of content similarity, such as similarity in prerequisite knowledge or difficulty, if a more informed definition of content similarity is used, the success of these methods is likely to grow.

Although there is a lot of value in knowing what content is similar to other content, what content should be considered similar is highly dependent on use case. This makes it a challenge for content creators to define the similarity in the content, as they don't necessarily know what their content will be used for. While some content is obviously similar, for example, two mathematics problems that are identical except for the numbers used in the problems, in other situations it is much more difficult, especially when content is being aggregated from multiple sources that may not even use the same metrics for prerequisite knowledge or difficulty.

Crowdsourcing offers a way to derive which content is similar to other content for specific use cases. Crowdsourced opinions on similar content can be gathered each time a new use case for similar content arises. By informing the workers, whose opinions are being crowdsourced, of the specific use case and requirements for similarity, the methods that rely on content being similar are more likely to be successful. However, crowdsourcing opinions on similar content poses some challenges as well. Before an online learning platform or intelligent tutoring system uses crowdsourced assertions of similarity, steps must be taken to assess the trustworthiness of workers whose opinions are being crowdsourced and ensure the truthfulness of the final assertions of similarity.

In this work we present a novel algorithm that both measures the reliability of the workers whose opinions are being crowdsourced, and determines, from these individual's opinions, what content is most likely to be similar to other content. To evaluate this method, we first simulated a wide range of conditions in which assertions of similarity were made, and compared the performance of our algorithm to the traditional alternative. We then performed a case study where teachers and college students were told to identify middle-school mathematics problems that evaluated a simi-

Ethan Prihar and Neil Heffernan "A Novel Algorithm for Aggregating Crowdsourced Opinions". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 547-552. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

lar skill set. The assertions of similarity collected from the case study were used to identify groups of similar problems and measure the reliability of each worker's assertions.

Ultimately, this work seeks to answer the following three research questions:

1. Can we exploit properties of community detection to more accurately form groups of content from crowdsourced opinions?
2. How does the resulting algorithm perform in a simulation study compared to the more traditional method?
3. How does the resulting algorithm perform in a case study using workers of various expertise to determine which mathematics problems are similar to each other?

2. BACKGROUND

2.1 Ensembling Crowdsourced Opinions

Identifying the truth from crowdsourced opinions is not a new problem. Most of the techniques employed to ensure the accuracy of crowdsourced opinions rely on ensuring that workers have sufficient knowledge of the subject matter. This can be done through testing workers before giving them tasks, tailoring tasks specific to their skill sets, recruiting high quality workers, and educating workers before assigning them tasks. This can also be done through encouragement with extrinsic motivators like money, promotions, or prizes, or intrinsic motivators like a sense of purpose, or by gamifying the crowdsourcing tasks [1].

While there are many methods to encourage individuals whose opinions are being crowdsourced to be accurate, this work is focused on how to validate the quality of individuals' opinions after their task is complete. Current methods for accomplishing this place the burden of validation back onto the workers. Having workers rank the quality of other workers assertions is one method of validation. Another common method for validation is to have multiple workers perform the same task and merge the output of each worker, either as an average or as a majority vote [1].

There are also more advanced ways of algorithmically validating crowdsourced opinions. Item response theory and latent factor analysis based models have out-performed majority voting based validation methods on tasks related to identifying facial expressions and answering questions about geography [6, 10]. These models also determine the quality of individuals whose opinions are being crowdsourced, which can be used to refine the pool of individuals used for future crowdsourcing tasks [6, 10]. The novel algorithm in this work also aggregates crowdsourced opinions while evaluating the quality of each worker.

2.2 Community Detection

The field of community detection is focused around determining groups of similar items from a network of connected items. This has many applications throughout mathematics, physics, biology, computer science, and social sciences. Many things can represented as a network, for example, interstellar objects, neurons, city streets, and social media can all be represented as networks of interconnected items [3]. Finding similar educational content can be framed as a community detection problem by representing educational content as a network in which items are connected by topic, difficulty, language, prerequisite knowledge, or, in the case of this work, opinions on similarity. Structuring the task of identifying similar educational content as a community detection problem allows for the use of various well-established community detection algorithms, such as hierarchical clustering. In hierarchical clustering, each item begins in it's own cluster. Then, clusters are merged based on the merge strategy and distance between clusters [5]. Hierarchical clustering was used in both the simulation and empirical study.

3. METHODOLOGY

3.1 Dynamically Weighted Majority Vote

The Dynamically Weighted Majority Vote (DWMV) method is our alternative to the traditional majority vote method for combining multiple crowdsourced opinions on tasks with binary outputs. The DWMV method calculates the weighted majority opinion for each task, then determines the weight of each worker by how closely their opinion agreed with the majority opinion. The closeness of a worker's opinion to the majority opinion can be determined with any function for comparing two vectors that results in a value greater than or equal to zero. For example, accuracy or Dice coefficient[2]. DWMV initializes all workers' weights to be equal at the beginning of the algorithm, and iteratively updates these weights until the weighted majority vote does not change between iterations. Once the weighted majority vote remains constant from one iteration to the next, the weights of the workers can be interpreted as a measure of confidence in each worker, and the final weighted majority vote can be used downstream in the same way the traditional majority vote would have been used. Algorithm 1 formally defines the DWMV algorithm. In Algorithm 1, the function $s(x, y)$ determines the closeness of worker i's opinion, $(B_{ij}[A_{ij} = 1])_{j=1}^{t}$, to the majority opinion, $(u_j[A_{ij} = 1])_{j=1}^{t}$. The algorithm requires a matrix A of response indicators, in which $a_{ij} = 1$ if worker i completed task j, and $a_{ij} = 0$ otherwise, and a matrix B of worker's responses to tasks, in which b_{ij} contains the binary response of worker i to task j. In Algorithm 1, vector u contains the final weighted majority vote for each task, and vector c contains the final measure of confidence for each worker, based on the similarity between the weighted majority votes and the individual worker's responses.

3.2 Simulation Study

To determine if DWMV had a positive impact on forming groups from crowdsourced opinion, a simulation study was performed to compare the DWMV method to the traditional majority vote method in a variety of conditions. Figure 1 illustrates the simulation process. In the simulation study, hierarchical clustering was used to form groups from simulated workers' opinions of item similarity aggregated using both the majority vote method and the DWMV method. Table 1 lists the different initial parameters and their values used in the simulation. Five trials of every possible combination of the values in Table 1 were simulated for a total of 37,500 simulation runs.

Algorithm 1 Dynamically Weighted Majority Vote

Require: $s(x, y)$: function for the similarity of two vectors
Require: w : number of workers
Require: t : number of tasks
Require: $A = (a_{wt})$: matrix of response indicators
Require: $B = (b_{wt})$: matrix of response values
$v \leftarrow (0)_{j=1}^{t}$ ▷ initialize with values different from u
$u \leftarrow (-1)_{j=1}^{t}$ ▷ initialize with values different from v
$c \leftarrow (1)_{i=1}^{w}$ ▷ start with equal confidence in all workers
while $u \neq v$ **do**

$$v \leftarrow u$$

$$u \leftarrow \left(\begin{cases} 1, & \text{if } \frac{\sum_{i=1}^{w}(c \odot B \odot A)_{ij}}{\sum_{i=1}^{w}(c \odot A)_{ij}} \geq \frac{1}{2} \\ 0, & \text{otherwise} \end{cases} \right)_{j=1}^{t}$$

$$c \leftarrow \left(s\Big((u_j[A_{ij} = 1])_{j=1}^{t}, (B_{ij}[A_{ij} = 1])_{j=1}^{t} \Big) \right)_{i=1}^{w}$$

end while

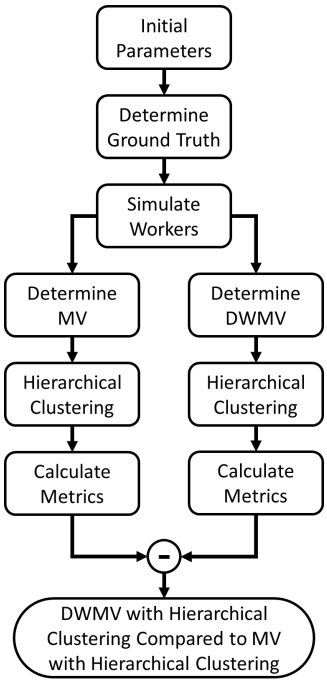


Figure 1: A flowchart of the simulation process, DWMV and majority vote were compared to each other through their use in community detection through hierarchical clustering.

Table 1: Simulation Parameters and Simulated Values

Parameter	Values
i	50, 100, 150, 200
g	5, 10, 15, 20, 25
w_{fp}	0.1, 0.2, 0.3, 0.4, 0.5
w_{fn}	0.1, 0.2, 0.3, 0.4, 0.5
p	20, 40, 60, 80, 100
d	0.25, 0.5, 0.75

The simulation began by randomly placing i items into g groups, where i and g are initial parameters of the simulation. Then the simulation crated ten workers. Each worker had a false positive rate and a false negative rate. These values were calculated separately to make the simulation more true to real life. In real life, it is not often that a worker would have an equal chance of incorrectly asserting that two items are or are not similar. The more likely case is that some workers think there is more similarity and other workers think there is less similarity between items than the actual similarity of items. The false positive and false negative rates of the workers were sampled separately for each worker from a uniform distribution in the range $[0, w_{fp}]$ and $[0, w_{fn}]$ respectively, where w_{fp} and w_{fn} are initial parameters of the simulation. Once the items were randomly placed in groups, and the error rates of the workers were randomly determined, a random p percent of all pairs of items were given to each worker, where p is an initial parameter of the simulation. Each worker then determined whether or not the items in each pair they received were similar to each other, taking into account their error rates.

Once all workers asserted whether or not each item pair they were given contained similar items, the majority vote and DWMV for the similarity of each item pair was calculated. The majority votes and DWMVs of item similarity were then used to form a network of item similarity, where each item is connected to every other item it was voted to be similar to. The majority vote network and DWMV network were both used to form groups through hierarchical clustering with Jaccard Index as the distance metric. Jaccard Index was used as the distance metric because Jaccard Index does not take into account true negatives [8]. Most items are not similar to each other, so a metric that takes into account true negatives would be over-inflated and not as informative in this context. After forming groups from the majority vote and DWMV similarity networks, the difference in accuracy, precision, and recall between the groups formed from the majority vote and DWMV similarity networks were used to determine if the DWMV method improved upon tradition majority vote.

3.3 Empirical Study: Similar Problems

In addition to a simulation, an empirical study was performed to compare DWMV to majority vote on a real crowdsorcing task. In this study, middle school mathematics teachers and college students were given 50 seventh grade mathematics problems from the Engage New York[1], Illustrative Mathematics[2], and Utah Middle School Math Project[3] curricula. Each worker was told to identify problems that evaluate similar mathematics skills. The workers' crowdsourced opinions of similarity were aggregated using both DWMV and majority vote, and then grouped using hierarchical clustering, with Jaccard Index as the distance metric with a threshold of 0.75. The resulting groups were then compared to a ground truth, provided by ASSISTments, an online learning platform [4], in the form of Common Core State Standards Mathematics Skill Codes[4], which each problem

[1]https://www.engageny.org/
[2]https://illustrativemathematics.org/
[3]http://utahmiddleschoolmath.org/
[4]http://www.corestandards.org/

was tagged with. These ground truth skill tags were determined by trained experts and the designers of the above stated curricula. The difference in accuracy, precision, and recall between groups formed with hierarchical clustering from DWMV and majority vote were again used to evaluate the quality of the DWMV algorithm.

4. RESULTS

4.1 Simulation Study

To compare the DWMV method to the traditional majority vote method, the difference in accuracy, precision, and recall as a function of w_{fp}, w_{fn}, i, g, and p, as described in Section 3.2, were calculated. The first positive takeaway from the simulation is that DWMV was almost always more accurate than majority vote, regardless of the simulation parameters. Only when the simulation had more than twenty groups or the maximum false negative rate of workers was 20% or less did DWMV not reliably out perform majority vote, but it did not significantly underperform either. At most, DWMV was slightly less accurate than majority vote when workers had very low false negative rates. Interestingly this increase in performance was not shared by both precision and recall. While recall followed the trend of accuracy and showed almost entirely positive improvements from using DWMV over majority vote, precision did not.

Another interesting finding is that all three performance metrics increased as both the maximum false negative rate and fraction of links seen by workers increased. This implies that as workers answer more problems, and become worse at correctly identifying when items are similar, the benefit of using DWMV over majority vote increases.

Overall, t-tests [9] showed that using DWMV led to a statistically reliable ($p < 0.001$) 0.18% increase in accuracy, a statistically reliable 1.78% ($p < 0.001$) increase in recall, but no statistically reliable ($p = 0.28$) change in precision. While small, these reliable improvements in accuracy and recall over the traditional majority vote method are an indication of the potential positive effects of transitioning to using DWMV instead of majority vote when aggregating crowdsourced opinion.

There were also some interesting differences in how different types of error affected the weights of workers as determined by the DWMV method. Figure 2 shows the average and 95% confidence interval of the DWMV weights of workers as a function of the workers' false positive and false negative rates. The false positive rate of the workers seems to decrease their weight in the final weighted majority vote of the DWMV method much more quickly than their false negative rate. A potential cause of this is that, in the simulated groups of similar items, there were far more pairs of items that were not similar to each other than there were pairs of items that were similar. For example, to have an equal number of items that are similar and not similar to each other, each item would have to be similar to half the items. The only way to facilitate that in the context of this simulation would be to have only two equally sized groups of items. In the simulation there were always at least five groups, and up to 25 groups of similar items, which caused most problems to not being similar to each other. Therefore, when a worker had a large false positive rate, there were more opportunities for them to make a mistake compared to a worker with a large false negative rate. Additionally, the large number of dissimilar problem pairs compared to the number of similar problem pairs caused workers with very low false positive rates to have higher weights than workers with equally low false negative rates, because workers with low false positive rates, regardless of their false negative rates, had much fewer opportunities to make a mistake. These findings suggest that the distribution of correct responses in crowdsourcing tasks affects which type of worker error has a larger impact on workers' weights in the DWMV method.

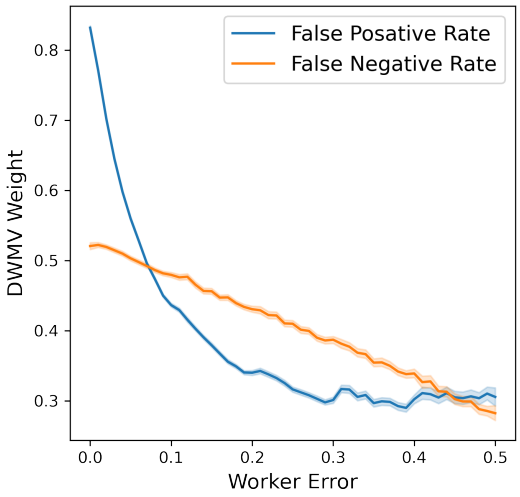


Figure 2: The average and 95% confidence interval of the DWMV weights of workers as a function of the workers' false positive and false negative rates.

4.2 Empirical Study: Similar Problems

In total, six teachers and four students completed the crowdsourcing task of grouping 50 seventh grade mathematics problems. Using each worker's assertions of similarity, the DWMV method and traditional majority vote were used to aggregate the opinions of the workers into a final network of similarity, which was then used to create groups of similar problems using hierarchical clustering. This is the exact same process that was used to form groups in the simulation study. Figure 3 shows the progressive iterations of DWMV. Iteration 1 shows the unweighted average of each workers assertions. The DWMV method's process of iterating between calculating a weight for each worker and calculating the weighted majority vote shifted the weighted average of workers assertions toward the ground truth similarity of problems. This convergence was present in the simulated example in Section 3.1 as well. The benefit of the DWMV method over traditional majority vote lies in this ability to converge towards ground truth. Figure 4 shows the weight of each worker as a function of their error rate. The cohort of middle school mathematics teachers performed much better overall than the cohort of college students. The average accuracy of the teachers was about 97% while the average ac-

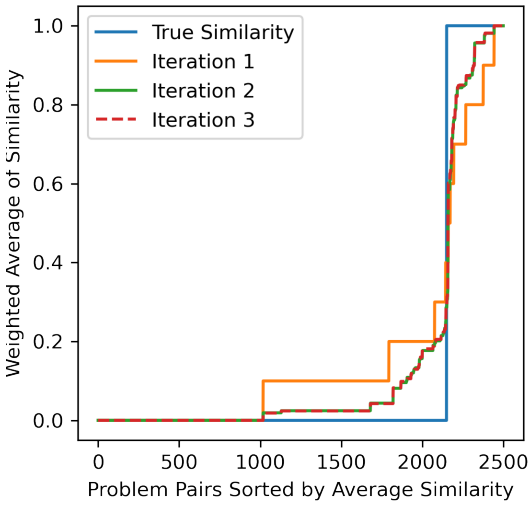


Figure 3: Progressive iterations of DWMV converging on empirical data.

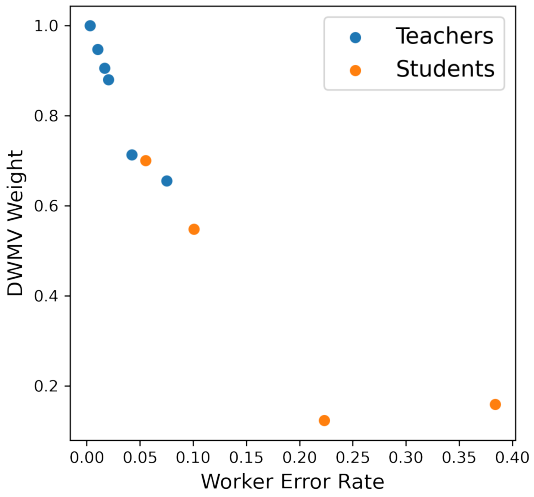


Figure 4: DWMV's confidence in each worker after the DWMV method converged.

curacy of the college students was only about 81%. Based on these weights, it is clear that the DWMV method valued the opinions of middle school mathematics teachers more than the opinions of college students, which is expected given the context and task. While, in this scenario, it might have been easy for a human in the loop to recognize that the teachers' opinions should be valued more, it will not always be the case that one group of workers is clearly more qualified than another group, and thus the DWMV method can help elucidate which workers are the most reliable.

Table 2 shows the difference in accuracy, precision, and recall between groups formed through hierarchical clustering from the assertions of similarity aggregated using DWMV and traditional majority vote. Similar to the simulation results, DWMV had the largest positive impact on recall, the second largest positive impact on accuracy, but no impact on precision. In this empirical study, both the traditional majority vote method and the DWMV method led to perfect precision, meaning all problems that were placed in groups together were similar to each other. However, traditional majority vote led to worse recall than DWMV. When tradition majority vote was used, three of the 50 problems were not placed in a group with any other problems, which is why the recall was so low. However, when DWMV was used, only one problem was not placed in a group of similar problems. This outlier problem, that neither traditional majority vote nor DWMV was able to correctly identify as similar to other problems in its group, had the following text:

> 22% of 65 is 14.3. What is 22.6% of 65? Round your answer to the nearest hundredths (second) decimal place.

Below are examples of problems in the same group as this problem, which were all correctly identified as similar to each other.

> Josiah and Tillery have new jobs at YumYum's Ice Cream Parlor. Josiah is Tillery's manager. In their first year, Josiah will be paid $14 per hour, and Tillery will be paid $7 per hour. They have been told that after every year with the company, they will each be given a raise of $2 per hour. Is the relationship between Josiah's pay and Tillery's pay rate proportional?

> To make a punch, Anna adds 8 ounces of apple juice for every 4 ounces of orange juice. If she uses 32 ounces of apple juice, which proportion can she use to find the number of ounces of orange juice x she should add to make the punch?

> A recent study claimed that in any given month, for every 5 text messages a boy sent or received, a girl sent or received 7 text messages. Is the relationship between the number of text messages sent or received by boys proportional to the number of text messages sent or received by girls?

Although all these problems are related to ratios and proportions, the other problems in the group with the outlier problem are longer word problems that do not explicitly use percentages. The teachers and students whose opinions were crowdsourced could have missed the connection due to the different wording in the problems, or they could believe that calculating percentages is a different skill than calculating proportions from word problems. Based on the differences between this single outlier problem and the other problems in its group, it is possible that the outlier problem was consciously excluded from its group and not simply an oversight.

The impact of using DWMV was larger in this empirical

Table 2: A comparison of majority vote to DWMV used to form groups of similar problems from crowdsourced assertions of similarity.

Metric	Majority Vote	DWMV	% Increase
Accuracy	0.987	0.997	1.054
Precision	1.000	1.000	0.000
Recall	0.903	0.977	8.228

study than it was in the simulation. In the simulation there was a larger than average improvement in accuracy and recall when the workers had very low false positive rates. Given that in this empirical study both sets of groups of similar problems had perfect precision, it is likely that the workers in this study had very low false positive rates, which likely contributed to why the positive impact of using DWMV instead of majority vote was larger in this empirical study than in the simulation as a whole. The results of this empirical study suggest that not only can DWMV out-perform traditional majority vote in simulations, but can also improve the recall and accuracy of groups of similar problems formed from crowdsourced opinions on content similarity in real-life scenarios as well.

5. CONCLUSION

Within online learning platforms and intelligent tutors, there is tremendous utility to knowing what content is similar to other content within the platform, but each application of similar content is likely to have different criteria for what is considered similar. Crowdsourcing opinions on the similarity of content is an accessible way for new applications to recognize similar content. However, crowdsourcing poses some difficulties, namely, how to identify reliable workers and properly aggregate opinions from multiple workers. This work has demonstrated the ability of the Dynamically Weighted Majority Vote method, a novel algorithm for aggregating crowdsourced opinion while rating workers, to accomplish those goals. DWMV has been shown, in both a simulation study and an empirical study, to lead to higher accuracy and recall that the traditional majority vote method on crowdsourcing tasks related to identifying similar content. In the simulation study, using DWMV before identifying groups of similar items through hierarchical clustering resulted in a statistically significant 0.18% increase in accuracy and a 1.78% increase in recall over using majority vote. The simulation study also revealed how the distribution of correct responses in the crowdsourcing tasks effects how the false positive and false negative rates of workers effects their weight in the DWMV method. In the empirical study, using DWMV before identifying groups of similar problems through hierarchical clustering resulted in about a 1% increase in accuracy and an 8% increase in recall over using majority vote, and provided perspective on the differences in accuracy between the expert middle school math teachers and the novice college students. Moving forward, when faced with the need to aggregate crowdsourced opinions, the learning science community can look to the DWMV method as an alternative to the traditional majority vote method. The DWMV method is a promising tool for increasing the reliability of crowdsourced opinion and, when paired with hierarchical clustering, identifying groups of similar content.

6. ACKNOWLEDGMENTS

We would like to thank multiple NSF grants (e.g., 1917808, 1931523, 1940236, 1917713, 1903304, 1822830, 1759229, 1724889, 1636782, 1535428, 1440753, 1316736, 1252297, 1109483, & DRL-1031398), as well as the US Department of Education for three different funding lines; a) the Institute for Education Sciences (e.g., IES R305A170137, R305A170243, R305A180401, R305A120125, R305A180401, & R305C100024), b) the Graduate Assistance in Areas of National Need program (e.g., P200A180088 & P200A150306), and c) the EIR. We also thank the Office of Naval Research (N00014-18-1-2768), Schmidt Futures, and an anonymous philanthropic foundation.

7. REFERENCES

[1] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys (CSUR)*, 51(1):1–40, 2018.

[2] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[3] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[4] N. T. Heffernan and C. L. Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4):470–497, 2014.

[5] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[6] P. Ruvolo, J. Whitehill, and J. R. Movellan. Exploiting commonality and interaction effects in crowdsourcing tasks using latent factor models. In *Neural Information Processing Systems. Workshop on Crowdsourcing: Theory, Algorithms and Applications*. Citeseer, 2013.

[7] D. M. Stenhoff, B. J. Davey, B. Lignugaris, et al. The effects of choice on assignment completion and percent correct by a high school student with a learning disability. *Education and treatment of Children*, 31(2):203–211, 2008.

[8] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[9] B. L. Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[10] J. Whitehill, T.-f. Wu, J. Bergsma, J. Movellan, and P. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in neural information processing systems*, 22:2035–2043, 2009.

Experimental Evaluation of Similarity Measures for Educational Items

Jaroslav Čechák
Masaryk University
Brno, Czech Republic
xcechak1@fi.muni.cz

Radek Pelánek
Masaryk University
Brno, Czech Republic
pelanek@fi.muni.cz

ABSTRACT

Measuring similarity of educational items has several applications in the development of adaptive learning systems, and previous research has already proposed a wide range of similarity measures. In this work, we provide an experimental evaluation of selected similarity measures using a large dataset. The used items are alternate-choice questions for the practice of English grammar for second language learners; the dataset contains thousands of items and over 10 million student answers. Our results provide warnings about the generalizability of results presented in EDM works: 1) the results vary significantly between knowledge components and 2) the size of available data is an important factor.

Keywords

item similarity, evaluation, generalizability

1. INTRODUCTION

Learning environments often contain thousands of educational items (questions, problems). A useful data mining contribution is to quantify the pairwise similarity of these items [9]. Such similarity measures have many applications. There are useful particularly for the management of the content, e.g., adding and deleting new items, preparing and revising explanations and hints, or deciding when to split knowledge components. Similarity measures can also be used in algorithms that guide the presentation of the content, e.g., in the presentation of error explanations, it may be useful to group similar items together; in sequencing items, we may want to avoid giving students two very similar questions in close succession. Item similarities may also be used for student modeling [6, 12].

Item similarity can be computed in many ways [9]; the basic two approaches are to use the item content data (e.g., the text of the question) and student performance data (e.g., the correctness of answers and response times). The content-based measures are, to a large degree, dependent on the specific type of data. The techniques based on student performance data are content-agnostic and widely applicable; the disadvantage is that they require (potentially large) student data. Previous research has proposed several specific measures [11, 7, 10].

In this work, we focus on the evaluation of previously proposed measures on a large and interesting dataset. The used items are alternate-choice questions for the practice of English grammar for second language learners (see examples in Table 1). The dataset contains thousands of items, which are categorized into knowledge components and difficulty levels. The items are alternate-choice questions, i.e., they consist of a stem, correct answer, and a single distractor. Items also have explanations, which are written in the Czech language. The dataset contains approximately 10 million student answers.

For this dataset, we evaluate various similarity measures and explore their relations. We focus particularly on the relation between performance-based measures and measures based on the text of explanations. We explore the issue of the sufficient size of data on student performance. In EDM research, this issue is often neglected; the performance of techniques is often studied using a fixed dataset ("all available data"). Our experiment shows that the studied methods are quite data-hungry; they require thousands of answers per item and the amount of available data seems to be more important than differences caused by choice of a measure (which is a type of result common with other machine learning applications [2, 4]). Experiments also show large differences in results between different knowledge components, even though all of these knowledge components come from a single domain (English grammar) and all the used items are of the same, simple format (alternate-choice questions). This result provides a warning about the generalizability of research results in educational data mining.

2. EXPERIMENTAL SETTING

In this section, we describe the data we used for experiments and the specific similarity measures.

2.1 Data

For the evaluation, we use data from the adaptive learning system Umíme anglicky, `umimeanglicky.cz`. The system contains various exercises for English grammar and vocabulary learning for second language learners (for Czech native speakers). We use only one type of exercise—alternate

Jaroslav Čechák and Radek Pelánek "Experimental Evaluation of Similarity Measures for Educational Items". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 553-558. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Table 1: Examples of items from the knowledge component *Present simple vs. present continuous*. For the sake of readability, explanations are given here in English; in the used data, they are in the Czech language.

item stem	correct	distractor	explanation
I _ to the gym once a week.	go	am going	When talking about periodical events, we use present simple tense.
I _ the film that we're watching.	hate	am hating	The verb to hate is not used in continuous form. We use present simple form instead.
I can't hear you! Everybody _ so loudly.	is talking	talks	When the activity is still in progress, we use present continuous tense.

choice question of the form fill-in-the-blank with two options (the correct answer and a distractor). The number of options is not crucial and our analyses could also be applied to questions with multiple distractors. The questions have explanations (in the Czech language).

The questions are divided into *item sets*. Each item set contains questions of similar difficulty from a single knowledge component. The system uses three difficulty levels. An example of an item set is *Present simple vs. present continuous, medium difficulty*, for which examples of questions are provided in Table 1.

Our dataset consists of 54 knowledge components divided into 68 item sets that in total contain 4 348 items. Some item sets share the same knowledge component, and they only differ in the difficulty of items. Concerning student performance, we use the answer (correct or incorrect) and response time (measured in milliseconds). We have 9 752 957 answers from 151 904 students.

Since details of data collection can often have a nontrivial impact on the results of the evaluation [8], we provide a basic description of the core aspects of system behavior that influence the collected data:

- In the system, students answer a sequence of items from a single item set in random order.
- The system uses mastery learning on the level of item sets. Students are motivated to answer a sufficient number of items correctly to satisfy the mastery criterion.
- The choice of an item set that a student solves can be done in a variety of ways: student free choice, assignment by a teacher (homework, assignment within a class), or recommendation by the system (based on past activity).
- The item sets differ widely in their difficulty. The samples of solvers may differ significantly for individual item sets (e.g., *Second conditional, hard* is solved by more advanced students than *Present simple tense, easy*).
- Items may move between difficulty levels ("design level adaptivity" [1]). This aspect may be important for some measures.

2.2 Similarity Measures

In our experiments, we use similarity measures that are variations on previously studied measures [9].

2.2.1 Measures Based on Item Content

One type of measure utilizes that available data about items. One possibility is to utilize item statements, e.g., to measure the similarity of item texts or match on options (the correct answer and distractor). In the case of grammar learning, this approach is hard to use: two questions that practice the same grammar rule can have completely different texts, answers, and distractors. We have performed preliminary experiments with various measures based on item text; these experiments showed very weak results. Therefore, we do not discuss these measures in more detail.

A more applicable content data are explanations. In the used dataset, each item has an associated explanation shown as feedback to students (particularly when they make a mistake). To quantify similarity based on explanations, we compute the text similarity of the explanations. To do so, we considered two common methods: Levenshtein edit distance [5] and Jaccard index.

Both methods compute the pairwise similarity of two explanations. Levenshtein edit distance operates at the character level and computes the minimal number of edits (character addition, removal, and substitution) required to transform one explanation into another explanation. Jaccard index only compares sets of words appearing in the two explanations regardless of their position. It is defined as

$$\frac{|E_1 \cap E_2|}{|E_1 \cup E_2|}$$

where E_1 is a set of words in one explanation and E_2 is a set of words in another explanation.

2.2.2 Measures Based on Student Performance

For computing similarity based on student performance, we consider two basic aspects: the correctness of answers and response times. These aspects are easy to collect and relevant for a vast range of items. In our experiments, we use similarity measures based on either of the two types of data and their combination.

Answer Correctness. The correctness of a student's answer is a simple binary indication of whether the student has answered an item correctly (selected the correct option

Table 2: Agreement matrix for items i and j. Values a, b, c, and d are numbers of students that answered both items in a particular way. For example, c is number of students that answered item i correctly but answered item j incorrectly.

$n = a+b+c+d$		item i correct	item i incorrect
item j	correct	a	b
	incorrect	c	d

$$S_p = \frac{(ad-bc)}{\sqrt{(a+c)(a+b)(b+d)(c+d)}}$$
$$S_c = \frac{(P_o - P_e)}{(1-P_e)}$$
$$P_o = \frac{(a+d)}{n}$$
$$P_e = \frac{((a+b)(a+c)+(b+d)(c+d))}{n^2}$$
$$S_{kl} = \frac{(ad-bc)}{(a+c)(c+d)}$$

in our case). Similarity measures based on the answer correctness then measure "agreement" between answers given by the same students to different items. This is best illustrated on an agreement matrix for two items i and j. There are only four possible ways a student can make binary responses to two items, as illustrated in Table 2. Similarity measures then differ in how exactly they compute the agreement from the individual components of the matrix. In our experiments, we use Pearson correlation coefficient (S_p), Cohen's Kappa (S_c) [3], and Kappa Learning [7] (S_{kl}).

Answer correctness measures can be extended by including a "second step" [9], i.e., computing similarity of similarities. In the first step, binary vectors of student answers for two items are compared to obtain the two items' similarity. The result is a similarity matrix with real-valued elements $s_{i,j}$ equal to the similarity of items i and j. The second step compares real-valued vectors $s_{i,*}$ and $s_{j,*}$ to obtain similarities of items i and j. In our experiments, we use Pearson-Pearson which is a Pearson correlation coefficient used in both first and second step.

Response Time. Response time is measured as the time it takes a student to answer the item (read the item statement and click on one of the options in our case). Student response times can vary due to external distractions during answering or even technical reasons like unreliable internet connection. To make the measure more robust, we opted to bin each item's response times into percentiles. The similarity of two items i and j is then measured as Pearson correlation coefficient of student response time percentiles vectors for items i and j.

Combined. Both correctness and response time can be combined to extract more bits of information. There are multiple ways to combine correctness and response time into a single score [9]. In our experiments, we use linear time transformation for correct answers as a combined score defined as $r = c \cdot max(1 - t/2\tau), 0)$ where $c \in \{0, 1\}$ is correctness, $t \in \mathbb{R}^+$ is response time, and τ is the median time for a given item. Similarities of items i and j are then Pearson correlation coefficient of score vectors for items i and j.

Table 3: Overview of all item similarity measures used in this study.

name	measure type	data used
Levenshtein edit distance	content	explanations
Jaccard index	content	explanations
Pearson corr. coef.	performance	correctness
Cohen's Kappa	performance	correctness
Kappa Learning	performance	correctness
Pearson-Pearson	performance	correctness
Response time percentile	performance	response time
Response time score	performance	correctness + response time

3. RESULTS

In this section, we present our findings. We use the explanations as "ground truth" for item similarity. The reasoning is that explanation describes the aspect of knowledge component that the item is practicing, and similar aspects are described in a similar way (e.g., same tense or conditional). This approach has its limitations, and it is heavily dependent on the quality of explanations. Not all explanations are necessarily ideal (different granularity between knowledge components, human errors), but it is a reasonable proxy.

For intuition behind the performed evaluation, Figure 1 provides an illustration using two knowledge components. The figure shows a PCA projection of items into plain based on the Pearson similarity measure that uses only the correctness of answers. The color of points is based on the explanations provided in the system. As we can see, these two approaches to measuring item similarity to a large degree agree—the points with the same color (similar with respect to explanations) are close to each other (similar with respect to performance). We now explore these relations in a more qualitative manner.

3.1 Relations Among Measures

Table 3 provides an overview of measures introduced in Section 2.2. Other measures can be defined in a similar fashion. An obvious question is whether they differ in any significant way or measure the same thing. To explore relations among measures, we first look at how much they are correlated. The correlation of two measures is computed as the Pearson correlation coefficient of item similarity matrices, each produced by applying item similarity measure to all pairs of items. A high correlation of two measures means that they generally agree on which pairs of items are similar.

Figure 2 shows correlations among measures based on performance and explanation averaged across all item sets. Both explanation-based item similarity measures are strongly correlated, and they also have comparable correlations with all performance-based measures. Therefore, it is not important

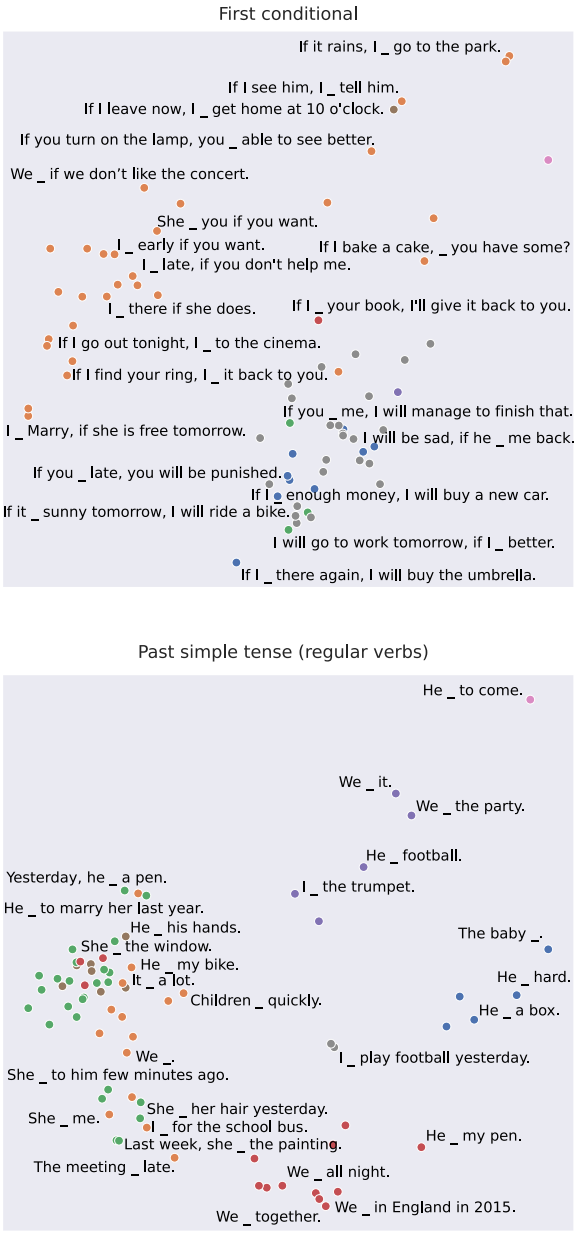


Figure 1: PCA projections based on measures using performance data (Pearson correlation). Points with the same color share the same explanations.

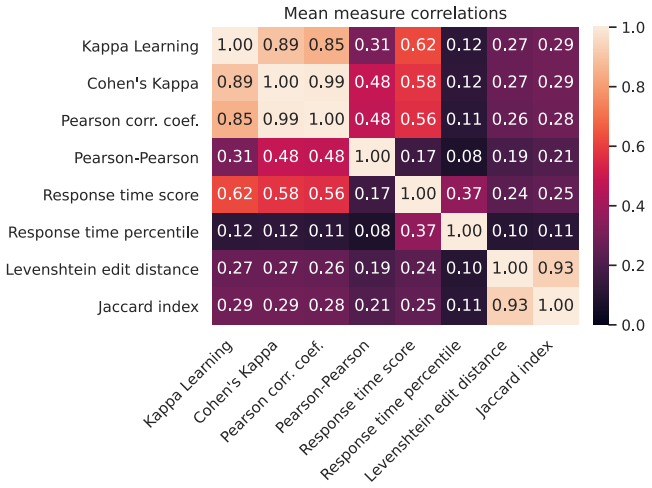


Figure 2: Heatmap of correlation among measures averaged across 68 item sets.

which one we choose as the ground truth for later experiments. This result is not surprising as both measures quantify text similarity, albeit in a different way.

Answer correctness measures Cohen's Kappa and Pearson behave almost identically, and their correlations across item sets are 0.96 or higher. The Kappa Learning measure also behaves similarly and has high correlations with both measures dropping below 0.75 only for one item set. When compared to explanation-based measures, all three measures achieve the same result. In most cases, it is not important which of the three we choose, and the amount of available data is a much more important factor (more details in Section 3.2). This result is in contrast to previous research [7], which argued that the Kappa Learning measure brings important improvement.

The second step similarity Pearson-Pearson has mostly the same or worse correlation with explanation-based measures compared to the previous three measures. It is related to Pearson and Cohen's Kappa, with correlation ranging from 0.3 to 0.8 for most item sets. The correlation with explanation-based measures is weaker compared to other measures using correctness. Thus for the used dataset, the second step does not seem useful. This observation is in contrast to previous research in another context [11].

The measures with response time do not provide any tangible benefits. When compared to explanation-based measures, they achieve either similar correlations in case of Response time score or very poor and mostly zero correlation in case of Response time percentile. A combination of answer correctness and response time in Response time score results in the best correlation for some item sets, but it is not significantly different on average. These results suggest that answer correctness might be a better indication of item similarity for our dataset.

3.2 Size of Data

Item similarity measures based on student performance are based on statistics of student performance data. All statistics need at least some amount of data to become stable and to start approximating the true statistical feature of the underlying data generating process. The question is then, how much data, i.e., answers per item, is required to obtain a good stable approximation?

In Figure 3, we have visualized the stability of performance-based measures in terms of correlation with the explanation-based measure. To simulate different numbers of answers, we have started with knowledge components with a sufficient amount of data and randomly subsampled each item's answers. We report correlation with an explanation-based measure; we report only the Jaccard index as it is highly cor-

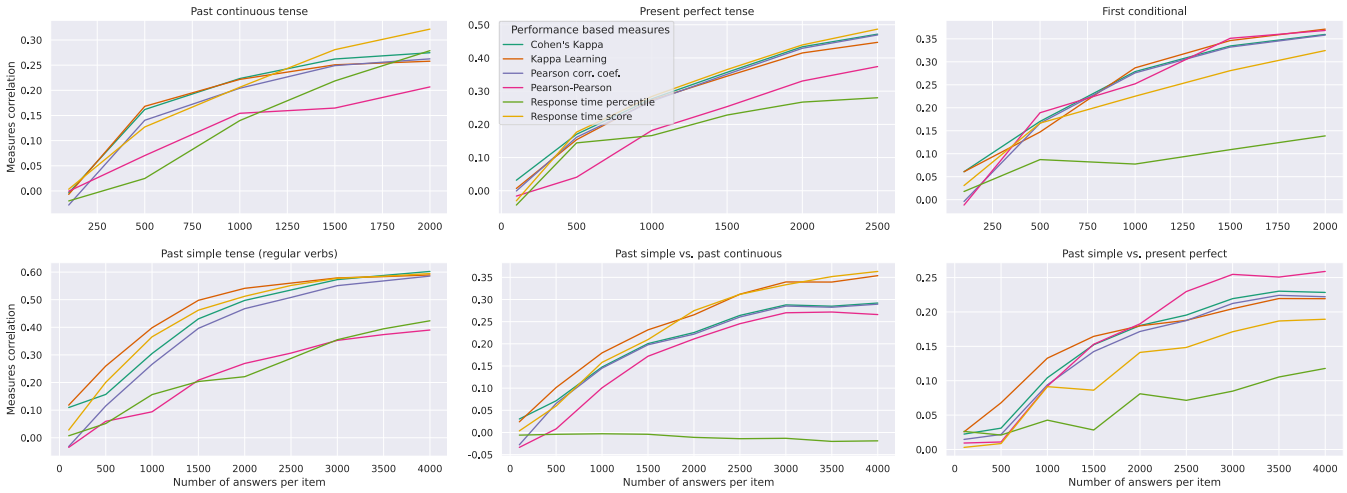


Figure 3: Correlation between performance-based measures and Jaccard index with an increasing number of answers per item across multiple knowledge components. Note that y-axis ranges differ between plots.

related with Levenshtein edit distance and has higher mean correlations with performance-based measures.

Figure 3 shows that performance-based measures are data-hungry. There are nontrivial differences in correlations until 2000 answers per item, and some improvement can be observed even for more data. The general shape of the curves is mostly similar across multiple knowledge components and final achieved correlations. There are a few changes in the relative ordering of measure, but these could be partly attributed to random noise for low data quantities. Different answer correctness measures have similar correlations regardless of data available. Response time score measures utilize more information from the data, and thus we expected them to converge faster. This, however, does not happen.

3.3 Differences among Knowledge Components

There are significant differences in the best achieved correlations among knowledge components. The best correlation achieved between any performance-based measure and explanation-based measure for a given knowledge component ranges from 0.06 to 0.67. Even if we filter out item sets with fewer than 2000 answers per item, the best correlation achieved are still between 0.25 and 0.67. Moreover, the ordering of performance-based measures in terms of achieved correlation with explanation measures differs between knowledge components. For example, Response time score with Levenshtein edit distance has the best correlation 0.61 for *Present simple tense* but the same pair has the worst correlation 0.06 for *Passive voice*. Therefore, the choice of knowledge component is more significant than the choice of similarity measures.

There is a multitude of factors causing these differences. We have identified some of these factors and give examples of their effect on correlations. The identified factors are features of the knowledge component, differences in student populations, and biases in data caused by the addition of content to the system.

Features of knowledge components describe how students use the knowledge component to answer an item. One such feature is how much the component is rule-based. There are more factual components, e.g., *Past simple tense of irregular verbs*, and more rule-based components, e.g., *Past simple tense of regular verbs*. In our data, more rule-based components achieve higher correlations on average. For example, *Past simple tense of regular verbs* achieved a correlation of 0.63 while *Past simple tense of irregular verbs* achieved only a correlation of 0.32.

The difference in student populations is especially important in systems that target a wider audience. The audience of item sets in our dataset range from grades 4 to 10, and thus the student population solving each item set differ. Simpler item sets for grades 4 to 7 achieve a better correlation of performance and explanation-based measures, while more advanced item sets for grades 8 to 10 achieve lower correlations.

Our dataset comes from a system that continuously evolves and has its content modified. These modifications also include the addition of new items among existing items. This poses a challenge for measuring similarity from performance data. Groups of items with varying amounts of collected data can make recently added items artificially different from the rest. For example, item set *Past tense: questions and negative* has 63 items with around 1700 answers per item and 20 newly added items with only around 800 answers per item. The best correlation between performance- and explanation-based measures rises from 0.3 to 0.36 when we filter out newly added items.

4. DISCUSSION

In this work, we have evaluated previously proposed measures for quantifying educational items' similarity based on students' performance. We have used a large dataset from a widely used learning system. The results provide important warnings for both practitioners and researchers.

Many educational data mining techniques require a large size of data for good performance. However, research papers often do not provide any indication of what size of data is good enough. Our results show that performance-based measures are data-hungry and may require upwards of 2000 answers per item before converging. Results reported on smaller datasets thus may be misleading in some aspects. Note that even a large university class would mean only around 200 answers per item which is still an order of magnitude smaller than the required 2000.

Another understudied issue is the generalizability of results across knowledge components. Our dataset is in many aspects very homogeneous: we consider only alternate-choice questions for English grammar. Nevertheless, there are non-trivial differences between the knowledge components (rule-based vs. fact-based, simple vs. advanced), and we have observed significant differences in results depending on the choice of a knowledge component. This observation raises a question of the generalizability of results reported on just a few knowledge components.

5. REFERENCES

[1] V. Aleven, E. A. McLaughlin, R. A. Glenn, and K. R. Koedinger. Instruction based on adaptive learning technologies. *Handbook of research on learning and instruction*, pages 522–560, 2016.

[2] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

[3] J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[4] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[5] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[6] Q. Liu, Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, and G. Hu. Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, 33(1):100–115, 2019.

[7] T. Nazaretsky, S. Hershkovitz, and G. Alexandron. Kappa learning: A new item-similarity method for clustering educational items from response data. *International Educational Data Mining Society*, 2019.

[8] R. Pelánek. The details matter: methodological nuances in the evaluation of student models. *User Modeling and User-Adapted Interaction*, 28(3):207–235, 2018.

[9] R. Pelánek. Measuring similarity of educational items: An overview. *IEEE Transactions on Learning Technologies*, 13:354–366, 2020.

[10] R. Pelánek, T. Effenberger, M. Vaněk, V. Sassmann, and D. Gmiterko. Measuring item similarity in introductory programming. In *Proc. of Learning at Scale*. ACM, 2018.

[11] J. Řihák and R. Pelánek. Measuring similarity of educational items using data on learners' performance. In *Educational Data Mining*, pages 16–23, 2017.

[12] S. Zhao, C. Wang, and S. Sahebi. Modeling knowledge acquisition from multiple learning resource types. *arXiv preprint arXiv:2006.13390*, 2020.

Analyzing Student Success and Mistakes in Virtual Microscope Structure Search Tasks

Benjamin Paaßen*
Insitute of Infomatics
Humboldt-University of Berlin
Berlin, Germany
benjamin.paassen@hu-berlin.de

Andreas Bertsch
Educational Technology Lab
German Research Center for Artificial Intelligence
Berlin, Germany

Katharina Langer-Fischer
Institute of Molecular and Cellular Anatomy
University of Ulm
Ulm, Germany

Sylvio Rüdian
Humboldt-University of Berlin & Weizenbaum Institute
Berlin, Germany

Xia Wang
Educational Technology Lab
German Research Center for Artificial Intelligence
Berlin, Germany

Rupali Sinha
Educational Technology Lab
German Research Center for Artificial Intelligence
Berlin, Germany

Jakub Kuzilek
Insitute of Infomatics
Humboldt-University of Berlin
Berlin, Germany

Stefan Britsch†
Institute of Molecular and Cellular Anatomy
University of Ulm
Ulm, Germany

Niels Pinkwart†
Insitute of Infomatics
Humboldt-University of Berlin
Berlin, Germany

ABSTRACT

Many modern anatomy curricula teach histology using virtual microscopes, where students inspect tissue slices in a computer program (e.g. a web browser). However, the educational data mining (EDM) potential of these virtual microscopes remains under-utilized. In this paper, we use EDM techniques to investigate three research questions on a virtual microscope dataset of $N = 1,460$ students. First, which factors predict the success of students locating structures in a virtual microscope? We answer this question with a generalized item response theory model (with 77% test accuracy and 0.82 test AUC in 10-fold cross-validation) and find that task difficulty is the most predictive parameter, whereas student ability is less predictive, prior success on the same task and exposure to an explanatory slide are moderately predictive, and task duration as well as prior mistakes are not predictive. Second, what are typical locations of student mistakes? And third, what are possible misconceptions explaining these locations? A clustering analysis revealed that student mistakes for a difficult task are mostly located in plausible positions ('near misses') whereas mistakes in an easy task are more indicative of deeper misconceptions.

*Corresponding author
†Shared senior authors

Benjamin Paaßen, Andreas Bertsch, Katharina Langer-Fischer, Sylvio Rüdian, Xia Wang, Rupali Sinha, Jakub Kuzilek, Stefan Britsch and Niels Pinkwart "Analyzing Student Success and Mistakes in Virtual Microscope Structure Search Tasks". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 559-565. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Keywords

anatomy education, clustering, item response theory, performance modeling, virtual microscopes

1. INTRODUCTION

Histology is a core subject that all medicine students have to pass in their studies. An important part of classic histology training is the microscopy course where students examine a large number of slides of human or animal tissue with an optical microscope in order to identify cellular structures with the aim of establishing structure-function relationships [21]. In recent years, more and more virtual microscopes (VMs) have been developed and integrated into teaching [21]. Such VMs reduce the need for resources (students only require a computer and a software), offer the opportunity to annotate slides with teacher notes, and enhance the student learning experience [21]. Prior work has provided numerous case studies of VMs being successfully integrated into anatomy education around the globe, e.g. [5, 6, 10, 13, 21, 22]. Moreover, several evaluation studies have shown that students using VMs perform at least as well as students using optical microscopes [11, 15].

To the best of our knowledge, no study to date has considered the educational data mining potential of VMs. For example, VMs enable us to record which slides students have seen, which areas on the slides they have focused on, etc. In this work, we consider the MyMi.mobile VM that is used in anatomy courses at two German universities [10]. In this VM, students can view a slide with expert annotations (exploration), and they can test their knowledge by either locating a structure in a slide (structure search; refer to Figure 1), or identifying the tissue sample and staining (diagnosis).

We analyze the performance of $N = 1,460$ students in structure search tasks with respect to three research questions:

RQ1: Which features predict student success?

RQ2: What are typical locations of student mistakes?

RQ3: What are possible misconceptions explaining these locations?

To answer RQ1, we analyzed the collected learning data with a generalized item response theory [2, 12] model, which consists of a difficulty parameter for each task, an ability parameter for each student, and four weights for additional features (see section 3.2). To answer RQ2 and RQ3, we employed a Gaussian mixture model [7] on the locations of mistakes and interpreted the resulting clusters with the help of domain experts. Our results can contribute to enhanced teaching quality in VM courses as well as establish interpretable models to analyze data from such courses.

In the remainder of this paper, we cover related work, our experimental setup, the results, and a conclusion.

2. RELATED WORK

Prior work on machine learning on virtual microscope data has focused on applications outside education. For example, major prior work has been done in training convolutional neural networks to solve classification tasks on microscope images such as detecting fluorescence on images [4]. Successful applications can assist anatomy experts in predicting carcinogens in human cells [23]. Due to the high accuracy of these models [1], they are helpful in cancer diagnostics.

Related to education, prior work of virtual microscopes can be roughly distributed into two categories. First, there are case studies describing how virtual microscopes were integrated into anatomy curricula and the requirements for successful integration, e.g. [5, 6, 10, 13, 21, 22]. Second, several studies have investigated whether students with optical microscopes have higher learning gain compared to students with a virtual microscope and found that this is not the case, e.g. [11, 15].

One of our research questions in this paper is to identify factors that are related to success in locating structures in a virtual microscope. Models that predict student success are a common topic of educational data mining research [3]. For example, Dietz-Uhler et al. [8] summarized which kind of data is often used to predict students success, classified into data gathered from the Learning Management System (e.g. clicks on resources) and performance data (e.g. feedback or grades, created by the instructor or respectively the system). Other papers use demographic data and prior success to predict success rates, e.g. [16]. Prior work has shown that, depending on the knowledge domain, different features have high importance to predict students' success. For example, Ramos et al. [20] found that hits in a discussion forum have high importance to predict students success. Yukselturk et al. [24] used a correlational research design and concluded that self-regulation variables have a highly statistically significant relation to learning success using interpretable methods. To our best knowledge, there is no prior work on success prediction in virtual microscopes. We want to close this research gap.

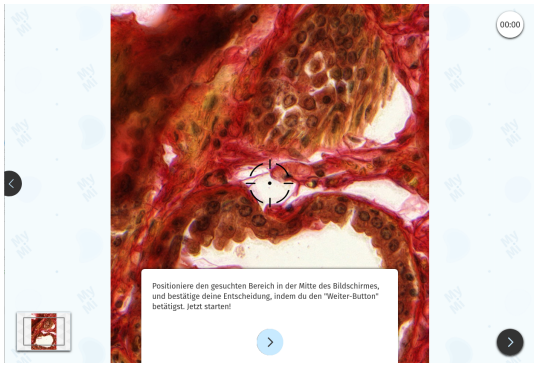


Figure 1: Screenshot of the MyMi.mobile structure search mode.

To do so, we turn to item response theory. Item response theory is concerned with modeling the probability of success of a student i at a task j via a logistic distribution over the difference between a student's ability parameter θ_i and a task's difficulty parameter b_j [2, 12]. Generalizations of this model include more parameters and other distributions [2, 12]. In this paper, we use the standard logistic distribution but include auxiliary parameters for features that capture student behavior.

To analyze the locations of typical mistakes, we perform a clustering analysis using Gaussian mixture models [7]. Clustering is a well-established technique in educational data mining [3], e.g. to identify groups of student solutions that may warrant similar feedback [9]. Our reasoning is similar: We wish to identify typical locations of mistakes in structure searches such that we have a reasonably sized set of representative locations that a teacher can inspect and for which feedback may be developed.

3. METHOD

3.1 MyMi.mobile VM and Dataset

The MyMi.mobile VM provides three modes: *exploration*, which shows expert annotations, *structure search*, where students need to locate a structure in a slide, and *diagnosis*, where students need to identify the slide and the stain. The structure search mode is shown in Figure 1. Students see a tissue slice and are supposed to move the field of view (by panning and zooming) until the crosshair is located over the correct structure. Then, they confirm their choice by clicking the arrow on the bottom right. As additional interface elements, students see an explanatory text at the bottom of the screen ("Position the area to be searched in the center of the screen and confirm your decision by pressing the 'continue-button'. Start now!"), a 'minimap' of the slide on the bottom left, and a timer on the top right. Students can select structure searches in any order from a list sorted alphabetically according to the slides (e.g. armpit, eye, colon)[1]. Students can attempt the same search as many

[1]The alphabetical ordering probably introduces an ordering bias. In particular, we observe that the two most attempted

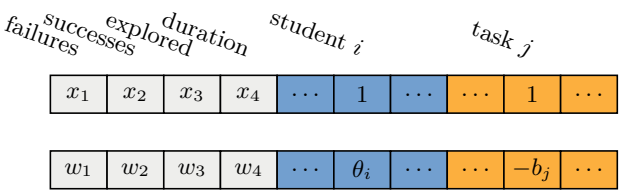


Figure 2: Illustration of the feature vector $\vec{x}$ (top) for an attempt of student i on task j, and the parameter vector $\vec{w}$ (bottom) of the item response theory model.

times as they want.

We consider a dataset of 19,525 structure search attempts by 1,460 students recorded in the summer term 2020 at two German universities. Most students were second semester undergraduate students of medicine, with some students from fourth semester dentistry (45) and molecular medicine in the second or fourth semester (39). Most students (817) did not attempt any structure search. Of the 643 who did, most attempted few structure searches (median 7) with some 'heavy users' making hundreds of attempts (mean 30.37, maximum 649). 68.19% of attempts were successful.

For the purpose of validating our model, we also asked four anatomical teachers using the VM to rate the difficulty of the 30 most attempted structure search tasks on the platform. The teachers received the following instruction: any structure search that at least 65% of students are expected to solve on their first try should be rated as 'easy'; any structure search between $40-65\%$ should be rated 'moderate'; and any structure search below 40% should be rated as 'difficult'. These boundaries were chosen based on the actual success rates of students: 10 of the tasks had an actual success rate over 65%, 10 had an actual success rate between $40\%-65\%$, and 10 had an actual success rate below 40%.

3.2 Item Response Theory

In order to investigate RQ1, we trained a generalized item response theory model implemented via logistic regression. In particular, we pre-processed each structure search attempt to be represented as a 1,859 dimensional, highly sparse feature vector (see Figure 2). The first four dimensions (gray) contain auxiliary features, namely: 1) How often has the student failed on the same structure search? (*failures*) 2) How often has the student succeeded on the same structure search? (*successes*) 3) Has the student already seen the same slide in the exploratory mode? (*explored*), and 4) How many minutes has the student spend on the current structure search? (*duration*). The next 1,460 dimensions (blue) indicate which student made the attempt, i.e. feature $x_{4+i} = 1$ if the current attempt was made by student $i \in \{1, \ldots, 1460\}$ and 0 otherwise. The remaining 395 features (orange) indicate which task the attempt was made on, i.e. feature $x_{1464+j} = 1$ if the current attempt was made on task $j \in \{1, \ldots, 395\}$ and 0 otherwise.

structure searches are both on the alphabetically first slide.

Our model, then, has the form

$$P(1|\vec{x}) = \frac{1}{1+\exp(-\vec{w}^T \cdot \vec{x})} \tag{1}$$
$$= \frac{1}{1+\exp(b_j - \theta_i - w_1 \cdot x_1 \ldots - w_4 \cdot x_4)}$$

where $\vec{x}$ is the sparse feature vector of an attempt and $\vec{w}$ is the parameter vector (see Figure 2). Note that we obtain a classic IRT model if the first four features x_1, x_2, x_3, and x_4 are 0. We used the implementation of logistic regression from the scikit-learn library [19].

3.3 Clustering

To investigate RQ2 and RQ3, we applied clustering on the locations of mistakes. More specifically, we used a Gaussian mixture model with K components, which approximates the probability density over locations (x, y) of mistakes in an image as

$$p(x,y) = \sum_{k=1}^{K} \mathcal{N}\big((x,y)\big|\vec{\mu}_k, \mathbf{\Sigma}_k\big) \cdot \pi_k, \tag{2}$$

where $\mathcal{N}((x,y)|\vec{\mu}_k, \mathbf{\Sigma}_k)$ denotes the 2D Gaussian density with mean $\vec{\mu}_k \in \mathbb{R}^2$ and covariance matrix $\mathbf{\Sigma}_k \in \mathbb{R}^{2\times 2}$; and where $\pi_k \in [0,1]$ is the prior for the kth Gaussian component. Compared to other clustering algorithms, Gaussian mixtures have at least two advantages. First, they can deal with non-spherical clusters by adjusting the covariance matrix accordingly. Second, they provide a probability density of the data. Moreover, they remain fast to train with an expectation maximization scheme [7]. We use the scikit-learn implementation of Gaussian mixtures [19]. To select the optimal number of components K, we use the Bayesian information criterion [18].

4. RESULTS AND DISCUSSION

In this section, we present the results of our experiments. We begin with the teacher difficulty ratings, then continue with the item response theory model (regarding RQ1), and conclude with the clustering analysis (regarding RQ2 and RQ3).

4.1 Teacher difficulty ratings

As the result of our teacher survey, we obtained difficulty ratings ('easy', 'moderate', or 'difficult') for the 30 most attempted structure search tasks. We observe that the teachers agreed moderately. On average, the Kendall τ for pairwise agreement is 0.4 and the overall Krippendorff's α is 0.44. To enhance reliability, we consider the average rating of each task in the subsequent analysis. On average, teachers ranked most tasks as 'easy' (about 55%), fewer as 'moderate' (just below 35%), and very few as 'difficult' (about 10%; refer to blue bars in Figure 3). Recall that, according to actual success rate, all blue bars would have height 1/3. This indicates that teachers tended to underestimate the actual difficulty, which may be an instance of the 'expert blind spot', i.e. the phenomenon that experts may fail to imagine the difficulties of novices [17]. We will use the teacher ratings as reference to further validate our item response theory model below.

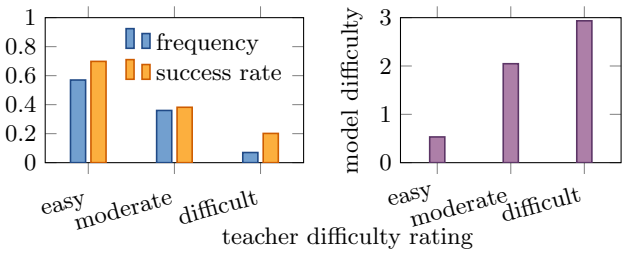


Figure 3: Left: The frequency (blue) and the mean actual success rate (orange) of tasks rated as easy, moderate, or difficult by teachers. Right: The average difficulty parameter assigned by the model to tasks rated as easy, moderate, or difficult by teachers.

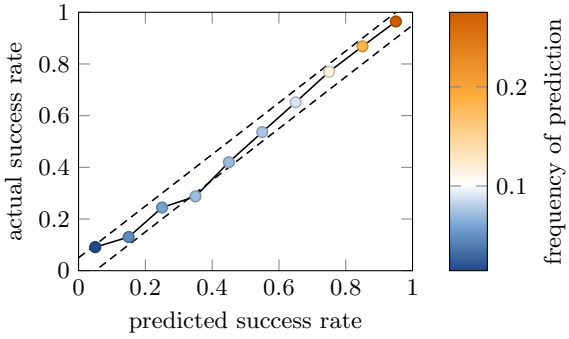


Figure 4: Calibration plot of the IRT model. Dashed lines indicate bin width. The color indicates how full each bin is.

4.2 Factors of student success

In order to investigate which factors contribute to student success (RQ1), we trained an item response theory model (refer to Section 3.2) on our data.

Model validation To validate the model, we performed three analyses. First, we performed a 10-fold cross-validation over attempts, yielding 80.19% ± 0.13% training accuracy and 77.73% ± 1.83% test accuracy on average ± standard deviation. Because our data is imbalanced (with less failures than successes), we also considered AUC (0.86 ± 0.001 in training and 0.82 ± 0.02 in test), and F1 score (0.69 ± 0.003 in training and 0.66 ± 0.024 in test with a test precision of 0.73 ± 0.06 and a test recall of 0.60 ± 0.03). All measures indicate good generalization from training to test set. For the remainder of this section, we consider a model trained on all data.

Second, we assessed model calibration. *Calibration* means that the predicted success probability of a student corresponds to the actual success rate [14]. To analyze this, we aggregated data into bins according to the predicted success probability (each bin had a width of 10%) and then computed the actual success rate within each bin. Figure 4 shows the corresponding calibration curve, where the dashed lines indicate the width of each bin in the analysis. Given that the curve remains within the dashed zone, we conclude that our model was well-calibrated. Most predictions

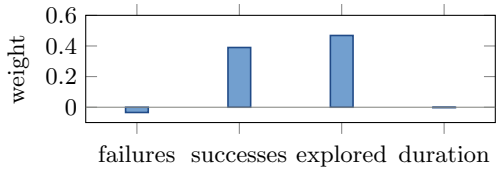


Figure 5: The scaled weights of auxiliary features.

(27.5%) were in the 90% – 100% bin (orange dot), i.e. our model predicted successful attempts with high confidence.

Third, we compared the difficulty parameters of our model with the human ratings from Section 4.1. Figure 3 (right) displays the average difficulty parameter assigned by the IRT model for each difficulty class. We observe that tasks rated as more difficult by teachers were also rated as more difficult by the model. Tasks rated as 'easy' by the teachers have a mean difficulty parameter of 0.5, tasks rated as 'moderate' have a mean difficulty parameter of 2, and tasks rated as 'difficult' a mean parameter of 3.

Overall, we note that the model is reasonably accurate, well-calibrated, and agrees with teacher ratings of difficulty.

Factors to Success Next, we inspect the weights of our model to infer which features are predictive of student success. To make the weights comparable, we normalized the auxiliary features to the same scaling as the binary features.

Regarding auxiliary features (Figure 5), we observe that the number of prior failures had a low negative weight, i.e. it is not predictive of student success. This is likely explained by the design of the MyMi.mobile VM. On a failure, students only learned that they were wrong but not where the right answer might be. This ensures that students can not get the right answer by trial and error. Attempt duration also had a low negative weight. This may be because duration is an ambiguous feature. Students may take longer both for productive reasons – e.g. inspecting the slide in more detail to validate the image against the definition of the structure – and unproductive reasons – e.g. being distracted. Accordingly, duration may not provide predictive information either way.

By contrast, we obtained positive scaled weights for the *successes* (0.39) and *explored* (0.47) features. The explanation for the former is obvious: If you have found the correct solution for the task once, chances are you memorized the location and can find it again. An explanation for the latter is that having seen an annotated example of the structure helps to find another instance of it in a structure search. That being said: We can not make causal inferences in this model. It is also possible that students who are more likely to succeed for other reasons are also more likely to consult the exploratory slides. On the other hand, we account for a general underlying student ability via the student ability parameter (Figure 6).

We observe that student ability parameters vary in the range from −1.97 to 1.55 and most parameters are clumped around

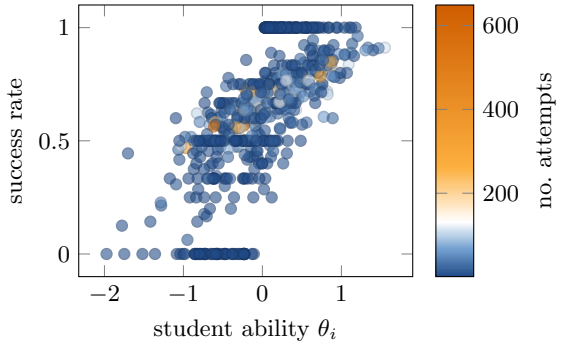


Figure 6: The success rate vs. the student ability of structure searches. Each dot represents a student. Color indicates the number of attempted structure searches by a student.

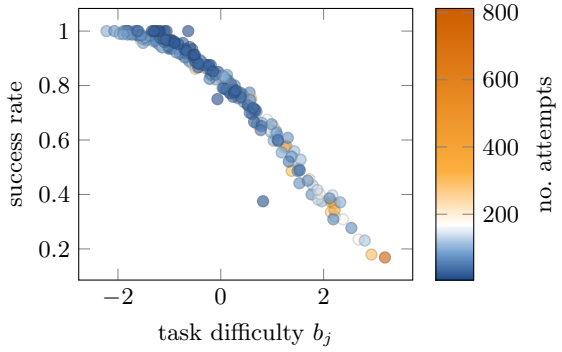


Figure 7: The success rate vs. the task difficulty of structure searches. Each dot represents a task. The heatmap colors indicate the number of attempts of a given task.

0 ± 0.54 (Figure 6). We also observe that the correlation between the ability parameter and actual success rate is relatively weak (Kendall $\tau = 0.52$). To further investigate the role of student ability, we performed another 10-fold cross validation over students instead of attempts, i.e. we tried to generalize to students that the model had never seen before and who thus had an ability parameter of 0. In this setting, we still obtained an average training accuracy of $80.19\% \pm 0.13\%$ and an average test accuracy of $77.93\% \pm 1.83\%$, indicating that the student ability parameters contributed little to an accurate prediction. We have two possible explanations for this finding: First, it may just be that the underlying 'true' student ability is relatively uniform because almost all students were in the same semester at the same two universities. Second, student ability may change during usage of the microscope, such that a single parameter may not be able to capture student ability particularly well.

Finally, we find that the task difficulty had the clearest relation to success compared to the other features. As shown in Figure 7, parameters range from -2.22 to 3.19 (mean 0 ± 1.19) and anti-correlate very well with the actual success rate (Kendall $\tau = -0.91$). This indicates that tasks had a roughly consistent difficulty across students. It also explains how our IRT model generalized well to new students.

In summary, we observe that prior success on the same task, having seen the corresponding exploratory slide, and task difficulty were most predictive of student success, whereas student ability was only moderately predictive and prior failures as well as duration were not predictive.

4.3 Typical mistakes

To investigate RQ2 and RQ3, we consider the two most attempted structure search tasks, namely searching for the nucleus of a myoepithelial cell and searching for an apocrine gland in human armpit tissue (refer to Figure 8 left and right, respectively).

The myoepithelial cell search (Figure 8, left) was the hardest task in the whole dataset with only 16.87% correct guesses (shown as green dots), with a difficulty parameter of 3.19, and unanimous consent of all four experts that it is difficult. Figure 8 (left) illustrates why the task is difficult: The correct regions (in green) are small and hard to spot.

By contrast, the slide for the apocrine gland task (Figure 8, right) exhibits many and large correct regions. Accordingly, 57.72% of guesses were correct (green dots), the model assigned a lower difficulty rating (1.28), and all experts agreed that this task is easy.

To identify typical mistakes, we trained a 10-component[2] Gaussian mixture model to cluster all the mistake locations (shown as blue dots). The cluster means are plotted as orange shapes in Figure 8. Interestingly, most clusters for the myoepithelial cell search task, namely the orange squares in Figure 8 (left) could plausibly be cell cores of myoepithelial cells. The bottom-most orange diamond is also located near a correct region. Only the remaining orange diamonds are clearly wrong because they are not located at cell cores. Generally, many students seemed to have a correct understanding of the structure to be found but failed to spot unambiguously correct locations.

By contrast, the cluster means for the apocrine gland search (Figure 8, right) indicate deeper misconceptions. All cluster centers are clearly wrong. More specifically, the diamond in the bottom right corresponds to an eccrine instead of andocrine gland, and the center diamond corresponds to a broken structure.

In both tasks, we can use cluster centers as a tool to find typical misconceptions that need to be discussed in class.

5. CONCLUSION

In this paper, we investigated three research questions regarding structure search tasks in virtual microscopes, namely 1) Which features predict student success? 2) What are typical locations of student mistakes? 3) What are underlying misconceptions explaining these locations?

[2]We observed that only little improvement in Bayesian information criterion could be achieved for more than 10 components. We also observed that 10 components were sufficient such that some components ended up unused in Figure 8. For other slides, different numbers may be needed.

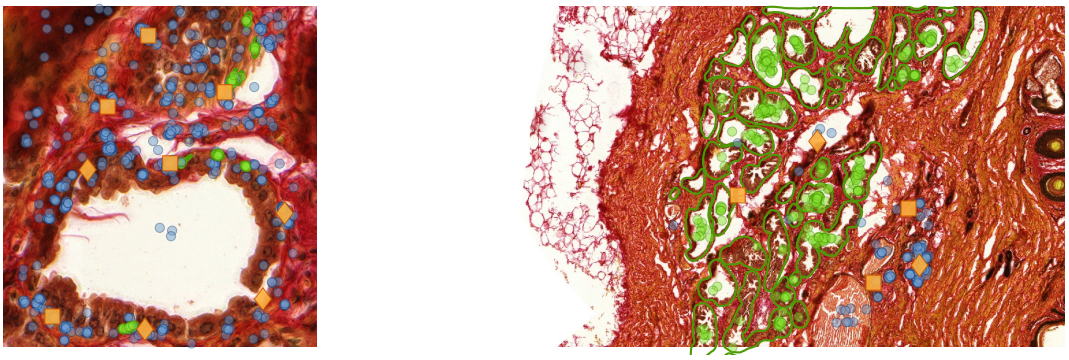


Figure 8: Students' correct (green) and wrong (blue) guesses on structure searches for myoepithelial cell cores (left) and apocrine glands (right). Correct structures are outlined in green. The centers of mistake clusters are orange shapes.

To answer the first question, we trained a generalized item response theory (IRT) model, obtaining 77% accuracy and 0.82 AUC in 10-fold cross-validation as well as solid calibration. Of the features considered, we found that task difficulty was particularly predictive of student success and the obtained difficulty parameters aligned well with actual student success rates and expert ratings. We observed less predictive value of student ability, illustrated by the fact that IRT models could generalize without loss of accuracy to new students. Moreover, prior success on the same task and having seen an annotated version of the same histological slide were predictive of success, whereas prior failures and duration spent on the task were not. This is interesting because it suggests that time stamps could be removed from the data, enhancing the privacy of the system.

Regarding the second and third research question, we applied clustering on mistake locations and interpreted the cluster centers in terms of misconceptions that may have led students to wrongly click at these locations. Such misconceptions can then be discussed in class to improve students' learning, or can be used to provide adaptive feedback in the virtual microscope tool.

Overall, this work represents the first step towards educational data mining on virtual microscope data with results that can be used to improve virtual microscope education, e.g. by ordering structure searches according to difficulty, by discussing typical misconceptions in class, and by enhancing annotations. Further work remains to be done, though. In particular, more features should be included to both enhance accuracy and find educational interventions that support student performance (like the exploratory view). Further, one could include relations between tasks in the model, thus identifying tasks that share an underlying skill, and extend the analysis to more advanced knowledge tracing methods. Finally, convolutional neural networks could be utilized to generalize teacher annotations and to identify regions of images that are easy to confuse with a structure to be searched.

6. ACKNOWLEDGMENTS

BP has been supported by the German Research Foundation (DFG) under grant number PA 3460/2-1, SR has been supported by the German Federal Ministry of Research (BMBF) under grant number 16DII127, and SB has been supported by grants from the Ministerium für Wissenschaft, Forschung und Kunst (MWK) Baden-Wuerttemberg, and by the Medical Faculty of the University of Ulm.

7. REFERENCES

[1] I. Anagnostopoulos and I. Maglogiannis. Neural network-based diagnostic and prognostic estimations in breast cancer microscopic instances. *Medical and Biological Engineering and Computing*, 44:773–784, 2006.

[2] F. Baker. *The basics of item response theory*. ERIC Clearinghouse on Assessment and Evaluation, College Park, MD, USA, 2001.

[3] R. S. Baker and K. Yacef. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining*, 1(1):3–17, 2009.

[4] R. Brent and L. Boucheron. Deep learning to predict microscope images. *Nature methods*, 15:868–870, 2018.

[5] L. David, I. Martins, M. Ismail, ..., and C. Carrilho. Interactive digital microscopy at the center for a cross-continent undergraduate pathology course in Mozambique. *Journal of Pathology Informatics*, 9(1):42, 2018.

[6] F. R. Dee. Virtual microscopy in pathology education. *Human Pathology*, 40(8):1112 – 1121, 2009.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38, 1977.

[8] B. Dietz-Uhler and J. E. Hurn. Using learning analytics to predict (and improve) student success: A faculty perspective. *Journal of interactive online learning*, 12(1):17–26, 2013.

[9] S. Gross, B. Mokbel, B. Paaßen, B. Hammer, and N. Pinkwart. Example-based feedback provision using structured solution spaces. *International Journal of Learning Technology*, 9(3):248–280, 2014.

[10] K. Langer-Fischer, D. Brandt, C. Braun, ..., and S. Britsch. MyMi.mobile - adaptives individualisiertes lernen in der mikroskopischen anatomie. In *Joint Annual Conference of the Medical Education Society (GMA), AKWLZ, and CAL*, 2019. German.

[11] B.-C. Lee, S.-T. Hsieh, Y.-L. Chang, . . ., and S.-C. Chang. A web-based virtual microscopy platform for improving academic performance in histology and pathology laboratory courses: A pilot study. *Anatomical Sciences Education*, 13(6):743–758, 2020.

[12] G. J. Mellenbergh. Generalized linear item response theory. *Psychological Bulletin*, 115(2):300–307, 1994.

[13] M. Merk, R. Knuechel, and A. Perez-Bouza. Web-based virtual microscopy at the rwth aachen university: Didactic concept, methods and analysis of acceptance by the students. *Annals of Anatomy - Anatomischer Anzeiger*, 192(6):383 – 387, 2010.

[14] M. E. Miller, S. L. Hui, and W. M. Tierney. Validation techniques for logistic regression models. *Statistics in Medicine*, 10(8):1213–1226, 1991.

[15] S. Mione, M. Valcke, and M. Cornelissen. Evaluation of virtual microscopy in medical histology teaching. *Anatomical Sciences Education*, 6(5):307–315, 2013.

[16] B. Naik and S. Ragothaman. Using neural networks to predict mba student success. *College Student Journal*, 38(1):143–150, 2004.

[17] M. J. Nathan and A. Petrosino. Expert blind spot among preservice teachers. *American Educational Research Journal*, 40(4):905–928, 2003.

[18] A. A. Neath and J. E. Cavanaugh. The Bayesian information criterion: background, derivation, and applications. *WIREs Computational Statistics*, 4(2):199–203, 2012.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, . . ., and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] C. Ramos and E. Yudko. "hits" (not "discussion posts") predict student success in online courses: A double cross-validation study. *Computers & Education*, 50(4):1174–1182, 2008.

[21] C. Schmidt, M. Reinehr, O. Leucht, N. Behrendt, S. Geiler, and S. Britsch. MyMiCROscope—intelligent virtual microscopy in a blended learning model at Ulm university. *Annals of Anatomy - Anatomischer Anzeiger*, 193(5):395–402, 2011.

[22] M. M. Triola and W. J. Holloway. Enhanced virtual microscopy for collaborative education. *BMC medical education*, 11(1):4, 2011.

[23] K.-H. Yu, C. Zhang, G. J. Berry, R. B. Altman, C. Ré, D. L. Rubin, and M. Snyder. Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features. *Nature Communications*, 7:12474, 2016.

[24] E. Yukselturk and S. Bulut. Predictors for student success in an online course. *Journal of Educational Technology & Society*, 10(2):71–83, 2007.

What you apply is not what you learn! Examining students' strategies in German capitalization tasks

Nathalie Rzepka
Hochschule für Technik und Wirtschaft Berlin
nathalie.rzepka@htw-berlin.de

Hans-Georg Müller
Universität Potsdam
Potsdam
hgmuelle@uni-potsdam.de

Katharina Simbeck
Hochschule für Technik und Wirtschaft Berlin
katharina.simbeck@htw-berlin.de

ABSTRACT

The ability to spell correctly is a fundamental skill for participating in society and engaging in professional work. In the German language, the capitalization of nouns and proper names presents major difficulties for both native and nonnative learners, since the definition of what is a noun varies according to one's linguistic perspective. In this paper, we hypothesize that learners use different cognitive strategies to identify nouns. To this end, we examine capitalization exercises from more than 30,000 users of an online spelling training platform. The cognitive strategies identified are syntactic, semantic, pragmatic, and morphological approaches. The strategies used by learners overlap widely but differ by individual and evolve with grade level. The results show that even though the pragmatic strategy is not taught systematically in schools, it is the most widespread and most successful strategy used by learners. We therefore suggest that highly granular learning process data can not only provide insights into learners' capabilities and enable the creation of individualized learning content but also inform curriculum development.

Keywords

Student strategies, Learning type, Online learning, German language, Spelling, Learning analytics

1. INTRODUCTION

The German language is known to be difficult to learn not only for nonnative speakers but also for native speakers who struggle with orthography [26]. However, a high degree of orthographic competence is crucial for successful communication with authorities and for professional success, as studies on employers and personnel selection show [21, 27].

One of the many peculiarities in the German language is capitalization. While nouns and proper names are generally capitalized, there are different linguistic perspectives on which words are considered nouns. Subsequently, learners can apply various redundant strategies to identify nouns.

Nathalie Rzepka, Hans-Georg Müller and Katharina Simbeck "What you apply is not what you learn! Examining students' strategies in German capitalization tasks". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 566-572. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Previous research has further indicated that these cognitive strategies for capitalization result in different patterns of errors that can be distinguished from each other [18]. While some learners consider the entire phrase when deciding whether to capitalize a word, others focus on only the word itself, especially the word ending, as an indication of the correct capitalization. Other learners use the words' meaning or take a pragmatic approach.

This paper aims to contribute to a better understanding of learners' cognitive strategies while processing capitalization tasks in German spelling courses. To this end, we use anonymized learning data on capitalization from the online platform orthografietrainer.net. The dataset consists of 9,647,385 single exercises completed by 30,658 users.

Identifying learners' cognitive strategies for capitalization tasks can enable educators and learning platforms to offer individualized help. Moreover, it can improve learning success by informing the implementation of personalized adaptive learning environments. Furthermore, comparing the predominant cognitive strategies in our large dataset to widely taught strategies in school can help inform future curriculum development. Previous studies of textbooks show that the set of rules taught in school contains semantic, morphological, and syntactic properties but almost completely lacks pragmatic strategy instruction [20]. Nonetheless, we found strong evidence that the pragmatic perspective is the major approach used by students of German.

In summary, we study the following research questions:

- RQ 1: Which cognitive strategies for capitalization are used by learners in grades 5 to 9?
- RQ 2: How does the use of capitalization strategies differ by grade level and gender?
- RQ 3: How do the predominant capitalization strategies used by learners compare to the strategies taught in school?

To answer the research questions, we proceeded as follows: The words used in the capitalization exercises on the online learning platform were manually one-hot encoded with 18 grammatical features associated with the four cognitive strategies for capitalization. In the next step, the four cognitive strategies for solving capitalization tasks were modeled as decision trees. Subsequently, the results of the four decision tree models were compared word by word with the solutions of more than 30,000 users.

2. RELATED WORK

2.1 Grammatical and cognitive approaches to German noun capitalization

The German orthographic system is complex and difficult to master. In contrast to other European writing systems, the difficulties relate less to spelling and more to the indication of grammatical structures. This can be illustrated by the capitalization of nouns, a

peculiarity of the German spelling system. Unlike in many other languages, in German, all nouns are capitalized. The ostensibly simple spelling rule that "nouns have to be capitalized" forces the speller to define precisely what is considered a noun and what is not. On closer examination, this question has a variety of very different possible answers.

On the one hand, there are many obvious nouns, such as people, places, things, and proper nouns. However, beyond that, every part of speech in German can be formally or functionally transformed into a noun. This is sometimes recognizable by a change in suffixes (cf. Ex. 1). In other cases, it can only be inferred from the syntactic context, for example, when articles or prepositions are added (cf. Ex. 2).

Ex. 1: fahren (V)	→	der Fahrer (N)
to drive	→	*the driver*
Ex. 2: fahren (V)	→	das Fahren (N)
to drive	→	*the driving*

The situation is further complicated by idiomatic expressions that formally contain a noun but that, from a pragmatic point of view, have lost their nominal characteristics (cf. Ex. 3). For instance, the supposed noun in Example 3 can still be formally complemented by an adjective, but this otherwise typical procedure for nouns is contrary to what a native German speaker would say. For this reason, the capitalization of such phrases is highly controversial in orthographic theory [5] and is a common source of error among students.

Ex. 3: im Allgemeinen	→ but not:	im *häufigen Allgemeinen
in general	→	*in *common general*

Consequently, all of the nouns in the first sentence of Jane Austen's "Pride and Prejudice" can be identified as nouns across several linguistic levels and work in both English and German:

Ex. 4: "*It is a truth universally acknowledged that a single man in possession of a good fortune must be in want of a wife."*

Ex. 5: *"Es ist eine allgemein anerkannte Wahrheit, dass ein Junggeselle im Besitz eines schönen Vermögens sich nichts mehr wünschen muss als eine Frau."*

Four of the nouns in the sentence occur with articles and in a typical syntactic environment for nouns. In addition, "man" and "wife" are identifiable as nouns by their concrete semantics. The words "truth" and "possession" are also marked morphologically since they were derived from the adjective "true" and the verb "to possess" with the help of a derivative ending.

The difficulties of coherent noun definition thus lie in the fact that a term may have a different extension depending on the linguistic perspective, although the semantic, morphological, syntactic and pragmatic perspectives agree in regard to a broad core of words. On the periphery, however, different perspectives lead to different conceptual boundaries and, consequently, to different orthographic decisions. The ground truth for what constitutes correct writing is therefore a mix between these different perspectives and defined by the Council of German Orthography [15].

The teaching of these different perspectives has been shown by an analysis of different textbooks [18]. The author found that capitalization is practically always introduced semantically. With the beginning of grammatical education in later primary school classes, morphological properties of nouns are added (especially the property numerus and some typical derivational endings), and articles as typical noun companions are introduced. At the secondary level, this knowledge is supplemented more systematically by further morphological and, especially, syntactic properties of the noun group (e.g., gender, case, other determiners besides the article). However, in all courses, noun identification is based exclusively on formal-grammatical grounds. Only one of the textbooks examined also refers to pragmatic properties of the nouns [20].

Müller [20] demonstrated that errors in capitalization correlate strongly with different linguistic perspectives. Thus, some learners are apparently guided more by semantic aspects and others more by morphological, syntactic or pragmatic factors. These findings provide a starting point for our study, in which we attempt to model the different perspectives on noun capitalization using learning analytics methods to test whether different learning types can be distinguished.

2.2 Cognitive strategies of capitalization

Very little literature exists on the differentiation of orthographic strategies. Theoretical models [16, 24] distinguish between lexical and syntactic approaches, which roughly correspond to semantic and morphological strategies on the one hand and syntactic and pragmatic strategies on the other. Studies on the success of both approaches [28] have been limited to very small corpora and have produced partly contradictory results. The proposal of a division into four individual strategies was made by [19], who also found initial indications of different error profiles on the basis of an empirical study.

According to our linguistic considerations, the investigation is based on four theoretically distinguishable capitalization strategies:

The semantic strategy capitalizes words that have a concrete meaning. This strategy is primarily taught in early elementary first grade: "Things that can be touched have to be capitalized."

Katze, Hand	→	but not: *nacht, *meinung,
cat, hand	→	night, opinion

The morphological strategy is to capitalize words that are classified as nouns because of the type of word and the word ending (word derivation):

Läufer	→	but not: (das) *laufen
Runner	→	(the) running

The syntactic strategy is to capitalize words that occur in a typical nominal syntactic environment, preferably in combination with attributes, articles or other determiners.

Die (totale) Dunkelheit	→	but not: *dunkelheit ängstigt mich.
The (total) darkness	→	Darkness frightens me.

The pragmatic strategy is to capitalize words that are used in the current discourse like a nominal unit, which does not apply to all nouns. Pragmatically proper nouns can be supplemented with attributes or substituted with pronouns, which is often not possible with nouns in fixed phrases.

Der Grund	→	but not: im *grunde
the ground	→	in the ground (saying for: "basically")

Typically, the use of several strategies leads to success. Furthermore, there are many words with which capitalization errors are made only very rarely, for example, articles, prepositions, and pronouns. There are only a few words where using only one strategy leads to the correct result, and these words are not representative in the German language. Nevertheless, learners apply the strategies to different degrees and thus arrive at different results.

2.3 Spelling error analysis

We identify the learners' use of the previously introduced cognitive strategies through the analysis of error patterns. The analysis of spelling errors can help in understanding students' cognitive approaches to assignments [2]. Many studies use spelling error analyses to gain knowledge about second language learners; for example, studies [3, 4] analyzed spelling errors of native Arabic speakers in English courses or programs. Others investigate special subpopulations, as the authors of [2, 23] did with dyslexic learners. In addition to differences between native language and foreign language learning and between subpopulations, there are different classification schemes for spelling errors. Some authors have used Cook's classification from 1999 [9], which differentiates between omission, substitution, addition, transposition and sound-based errors [3, 4]. There are major differences between writing systems, and Abu-rabia [1] showed that these differences also affect spelling errors. For the German language, Landerl and Wimmer [14] used the "phoneme distance score" as a scoring method for spelling errors. Defior and Serrano [10] divided Spanish spelling errors into seven different classes of errors, which consist of substitutive spelling, partial spelling, random letters and nonorthographic spelling. Czech spelling errors were divided into phonological errors on the one hand and orthographic, morphological, grammatical, and lexical errors on the other hand [7]. The information gained about the learners can later be used in adaptive environments for different educational approaches to best address each student's abilities [11].

2.4 Learner-Level Adaptation

Adaptive learning environments aim to improve learning success by building personalized models of each student's knowledge, preferences and difficulties [6, 12]. The goal of such an adaptation is to individually optimize the learning path for each student [17]. This can lead to higher motivation, less overload and frustration, and, thus, better results [17]. Personalized adaptation to the student's needs can appear in a variety of forms, including task sequencing, intelligent solution analyses and problem-solving support [6]. The adaptations and the subsequent assessment of adaptive learning environments use a range of different data [8]. The parameters used most often in learner-level adaptation are parameters that refer to the user him or herself and his or her profile as a learner to optimize content [17, 22]. The learner profile consists, among other components, of the learner's behavioral pattern, learner preferences, cognitive traits or learning style as well as performance data [8, 17]. The learner's behavioral pattern can be analyzed by tracing his or her activities on an online platform. Learner preferences thus basically describe learners' preferred materials [22]. Another approach is to adapt a system based on learners' cognitive traits. These traits are their cognitive abilities, for example, their working memory capacity, abstraction ability or analysis ability [22]. There are various definitions of learning style. However, they all agree that there are different ways that learners experience learning [13]. Fang et al. [11] also differentiated between features of a learner's interaction with a system and individual differences between learners in terms of, for example, skill and knowledge.

All this information can be used by teachers to gain a better understanding of their students, leading to opportunities to adapt their teaching, materials or tests [13]. In addition, learners can be provided with appropriate materials and tasks that meet their needs. Finally, learning styles differ in terms of the sequencing of tasks [13]. The relationship between learning styles and the structure of the learning material has been investigated by, for example, the authors of [25], who found that students whose learning styles and multimedia preferences match the material in their online course have higher scores.

In the context of this article, we suggest using the information gained about cognitive strategies for capitalization to display matching tips on online spelling platforms and to evaluate the difficulty of an exercise task in terms of which strategy is used.

3. DATASET

3.1 Orthografietrainer.net platform

The learning platform orthografietrainer.net offers online exercises for improving German spelling skills, including exercises on capitalization, punctuation, and spelling. The platform provides immediate and extensive individual feedback, which is impossible in a classroom setting. The training platform is built based on the pedagogical assumption that spelling requires not only knowledge but also skills. Thus, the focus is not on the regular learning of rules but on repeated practice [18].

The platform offers material for three different user groups: teachers, students and guests. Teachers register themselves and their entire class. They assign appropriate tasks to their students, who work on the tasks. Teachers can view their students' results via a dashboard. Additionally, any interested person can log in as a guest and complete tasks and tests.

A special exercise form on the platform is the competence test, which determines competence levels in capitalization, punctuation and separated or combined spelling. Any identified knowledge gaps are visualized, and appropriate exercises are suggested. A pretest, an intermediate test and a posttest are available and show improvements made over time. For this study, we use only data from competence tests on capitalization, not regular training data, as the test's standardized structure allows for better comparison. Moreover, in competence tests, all sentences are new to users.

3.2 Description of the dataset

For this paper, anonymized, event-level competence test data from orthografietrainer.net from April 1, 2020 to November 17, 2020 are used. Each answer to a sentence corresponds to one record in our dataset. During the analyzed time period, schools in Germany, Austria and Switzerland were closed for several weeks due to the COVID-19 pandemic. In this period, 46,356 users visited the online platform and completed a total of 65,645 capitalization task sessions. When processing the tasks, nearly 50% of the sentences were answered incorrectly, which means that the answers each contained at least one mistake.

The platform was heavily used during the first wave of the COVID-19 pandemic. During the German school holidays in July and August, there was less practice; online training activity increased again in autumn.

The dataset contains information about the class level and gender of users. The German school system includes grades 1 to 13, with 1 being the youngest children and 13 being the oldest (Figure 1).

We decided to exclude all users in grades 1 to 4, as those learners are not well represented in the data set and the difficulty of the capitalization exercises is not adjusted for them. Students in grade 10 and above are also excluded. Older students who are still assigned capitalization exercises are well behind the average learning path and thus represent a marginal group that would bias the data. Most of the users are in grade 7. Our dataset contains slightly more girls (51%) than boys (49%).

Age	School	Class level
18	Secondary School (Second Phase)	13
17		12
16		11
15	Secondary School (First Phase)	10
14		9
13		8
12		7
11		6
10		5
9	Primary School (Sometimes extended to Grade 6)	4
8		3
7		2
6		1

Figure 1. German School System (simplified)

3.3 Using decision trees to replicate different cognitive strategies

The capitalization of German words depends on various grammatical categories, such as the beginning of sentences, word types and clauses. The 180 sentences in the competency test that deal with capitalization contain 2679 words that begin with either lowercase or uppercase letters. These 2679 words were manually categorized into 18 grammatical categories. After one-hot encoding of the labels, we obtained a data frame with dimensions of 2679 x 58.

It cannot be assumed that people use only one strategy; instead, it is likely that each person uses different manifestations of a variety of strategies. To be able to analyze the students' adoption rates of the strategies in regard to capitalization, we first needed to gain insight into how a student would process a word if he or she were only to use one strategy and had only one preferred learning type.

For this purpose, decision trees were used to replicate the four cognitive strategies by attributing to them only the grammatical features corresponding to the given strategy. Afterwards, the sentences from the competence tests were predicted by the decision trees and then validated to determine whether the user classified the words correctly or incorrectly in terms of capitalization. This provided us with the error profiles that would result if only one of the four strategies were used to decide on proper capitalizations. Table 1 shows the strategies and their grammatical features.

Table 1. Strategies with grammatical features

Strategy	Grammatical features
Syntactic	Clause, Article, 2nd person, Determinator, Is prefix, Attribute, Complement of a prepositional phrase, Beginning of sentence, Core nominal pronoun
Semantic	Concrete, Polite form, Semantic word type
Pragmatic	Theme-Rheme, Attributable, Proper name, as an attribute not separable from noun sequence
Morphological	Word type, Noun ending

In the decision trees, 77 % of the words were processed correctly by all four strategies. These are mostly words, where users make only a few mistakes, such as articles, pronouns, prepositions, and conjunctions. They are not interesting for further analyses, as they do not provide insights into differences between the strategies. The beginnings of sentences are also filtered out because they are a special case and cause bias in the data: in the structure of the exercises on the online platform, the beginnings of sentences are in upper case letters per default. Students rarely click on such words to change the letter to lower case. However, as the beginning of a sentence is a syntactical feature, only the syntactic strategy processes these words correctly. Keeping the sentence beginnings part in the dataset would lead to bias, as most users would have a high adoption rate of the syntactic strategy precisely because the sentence beginnings are correct by default.

Table 2. Percentage of correct words per strategy

Syntactic	Semantic	Morphological	Pragmatic
74,11%	32,14%	62,05%	79,69%

The distribution of the remaining 448 words shows that strategies have different success rates (Table 2). The semantic strategy only processes approximately 30% of the words correctly, while the syntactic and pragmatic approaches are much better. This is not surprising, as the meaning of a word is less informative for the determination of capitalization than its grammatical use in a sentence.

Table 3. Sample of a merged data set

Word ID	User ID	Success	Syntactic	Semantic	Morphological	Pragmatic
255	452	1	1	1	1	0
256	128	0	1	0	0	1
257	427	1	0	0	0	1

In the next step, user data and the results from the decision trees are merged. The resulting data frame contains a word processed by a user in each row. For each word, there is information on whether the user capitalized the word correctly and how the decision tree models processed the item. Table 3 shows a sample of the resulting data set. In total, there are 1,355,641 records from more than 30,000 users.

4. RESULTS

To answer the first research question "Which cognitive strategies for capitalization are used by learners in grades 5 to 9", we compare users' error profiles with the error profiles of the decision tree classifiers. That for, we calculated the percentage of answers that matched. The adoption rate was calculated by dividing the sum of matching responses by the sum of processed words for each cognitive strategy. In this calculation, we did not consider whether the word was capitalized correctly. Instead, the result expresses only whether the words were processed in the same way by a user and by one of the four models.

Table 4 presents the average adoption rate per strategy in percentages. The models implementing the syntactic, morphological and pragmatic strategies were in alignment with the users' answers for, on average, 65% to 72% of the words. However, the result for the semantic strategy matched only approximately 40% of users' answers.

Table 4. Adoption rate by strategy

Syntactic	Semantic	Morphological	Pragmatic
65,33%	39,92%	66,32%	72,27%

When interpreting the results, it must be considered that several strategies can be used simultaneously when answering a task. This is always the case if the word cannot be answered exclusively by one strategy. Thus, overall, the adoption rate is over 100%.

4.1 Success rates

Thus far, we have primarily discussed the adoption of the four capitalization strategies. Now, we will examine the successful application of strategies for determining correct capitalization. Figure 2 shows the correlation between the adoption rates and the success rates per strategy.

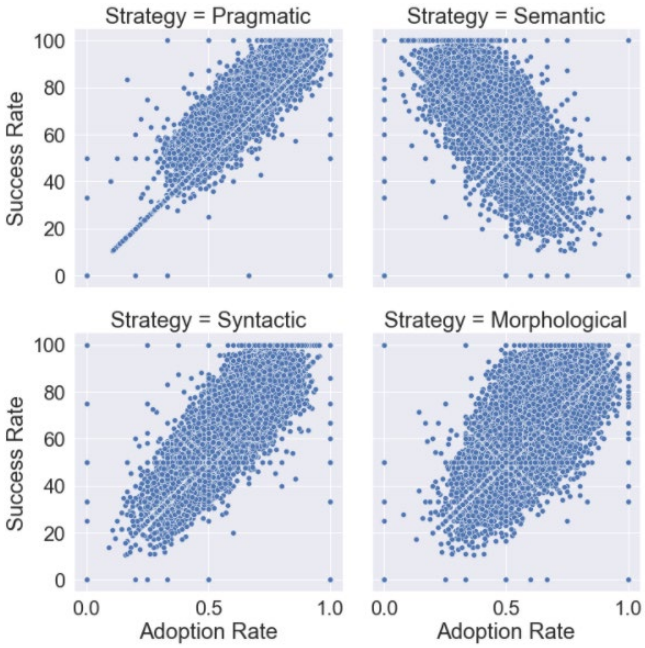


Figure 2. Correlation of success rate and adoption rate by strategy

The adoption of pragmatic, syntactic and morphological strategies led to increased success rates. The correlation is strongest for the pragmatic strategy. In contrast, the higher the share of words solved in agreement with the semantic strategy, the lower the success rate was. These correlations also exist when grade levels are considered in isolation. The success rates of the different strategies are also similar across grade levels.

The success distributed by class level and gender shows that students in higher grades tended to have lower success rates (Figure 3). While grades 5 to 7 had similar success rates, these declined from grade 8 onwards. The lowest success rates were found in grade 9. There is a very small difference between male and female success rates; however, in grades 7 to 9, male students correctly capitalized fewer words than female students did. It is possible that the data in these years reflect cognitive strategy shifts and corresponding temporary uncertainties.

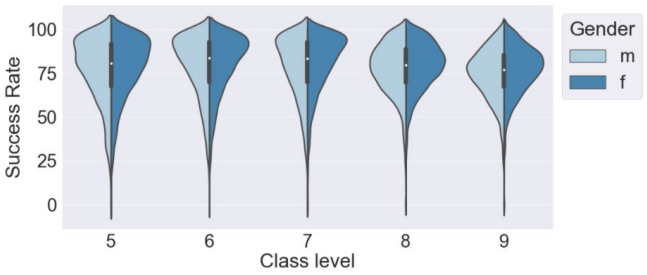


Figure 3. Success rates by class level and gender

4.2 Distribution by class level and gender

The second research question "How does the use of capitalization strategies differ by grade level or gender" is addressed by Figure 4. Looking at the distribution of the average adoption rate by strategy and grade level, we see that preferred strategies evolve over time and shift according to gender.

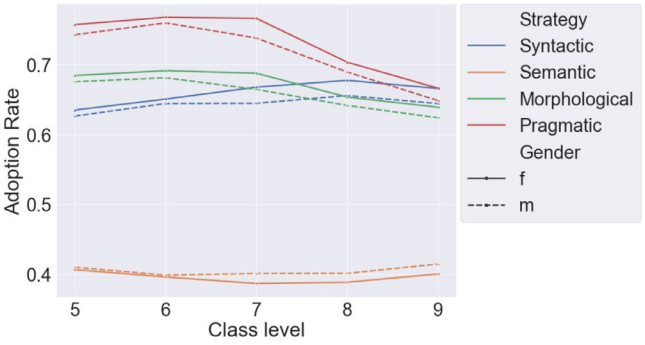


Figure 4. Distribution of adoption rates by class level and gender

The rate of adoption of the pragmatic strategy is very high from the beginning until it decreases sharply after grade 7 for girls and after grade 6 for boys. This is interesting, as the pragmatic strategy is the only strategy that is not explicitly taught in school even though it is very useful for determining correct capitalization (Figure 2). The pragmatic strategy is only surpassed in frequency by the syntactic strategy in grade 9, and the latter increases in use with every grade. Although the use of the syntactic strategy increases more for girls than for boys, in both cases, it ends up being on par with the pragmatic strategy.

Apparently, this reflects stronger grammatical skills among older students. Learners often start a second foreign language in grade 7 (usually Spanish or French), which increases the need for understanding grammatical concepts that are less explicit in their first foreign language, English. At the same time, usage of the morphological strategy also decreases from grade 7 onwards (as early as grade 6 for boys). These findings fit the students' learning biography, as grammatical instruction progresses from morphological to syntactic issues, and therefore orthographic instruction focuses on morphological strategies first. The adoption rate of the semantic strategy decreases until grade 7 but then increases again. This fits with the results regarding the success rate of the semantic strategy, which shows a weakening of knowledge from grade 8 onwards. The increase in semantic strategy use thus goes hand in hand with the students' lower success rates.

Looking at the differences in gender, we have already seen in Figure 3 that boys in grades 7 to 9 answer fewer words correctly than girls. If we now look at the use of the strategies by boys and girls

in Figure 4, we see that boys, especially in grades 7 to 9, use the semantic strategy more frequently, which is the least successful strategy and whose use correlates negatively with the success rate. Girls, on the other hand, use the other three strategies more frequently during this period, which correlate positively with the success rate.

In summary, we can again identify a difference between the semantic strategy and the other strategies. Even though the semantic approach is taught first, most learners do not adopt it for subsequent learning. The adoption rate of the pragmatic and morphological strategy decreases, while the syntactic strategy adoption rate increases. However, the pragmatic approach, which is rarely taught, is applied most frequently.

5. DISCUSSION

We used event-level learning data from an online spelling trainer to analyze cognitive strategies used by students for processing German language capitalization tasks. We built four decision tree classifiers to model capitalization strategies that use only syntactic, semantic, morphological or pragmatic features. As expected, as grammatical information in language is redundant, models often produce overlapping results. We compared the models' output to user error profiles. We found that the strategies are adopted to different degrees and that strong correlations—both positive and negative—between the adoption rates of strategies exist.

Furthermore, the distribution of adoption rates by grade level shows that strategies are represented among older and younger teenagers to different degrees. This variation by grade level is particularly interesting when compared to the rules taught at school, which answers the third research question "How do the predominant capitalization strategies used by learners compare to the strategies taught in school?". The first capitalization strategy taught at school is the semantic strategy: things that can be touched have to be capitalized. Even though this is taught first, students follow it only partly—and rightly so, as the semantic strategy is the least successful in determining correct capitalization. The pragmatic strategy (capitalizing a word if it occurs in a typical textual context for nouns), however, is the only one that is not taught explicitly in school. Nevertheless, this is the strategy with the highest adoption rate and with the highest success rate in our research. The syntactic strategy presupposes a deeper understanding of grammar than the semantic and pragmatic strategies and thus increases with grade level. Although the syntactic strategy and the grammatical knowledge required for employing it begin to be introduced in grade 5, it is only later that students apply it. This may be because students' actual understanding of German grammar increases when they begin learning a second foreign language in grade 7. Since many grammatical concepts are not present in English, a deeper engagement with grammar might only begin when students begin learning a second foreign language. This could lead to a different way of looking at spelling, which is then reflected in the use of the syntactic strategy. The use of the morphological strategy decreases over time as the use of the syntactic strategy increases.

When considering the success rates in combination with the adoption rates, it is particularly interesting that the semantic strategy adoption rate correlates negatively with success rate. This again shows that the teaching of the semantic strategy as the basic rule does not lead to success. The strongest positive correlation with the success rate is the pragmatic strategy adoption rate.

6. CONCLUSION

In this paper, we have contributed to three aspects of learning analytics. We have identified cognitive strategies of learners using error analyses, compared adoption rates and drawn conclusions for curriculum development from the results.

First, we were able to model cognitive strategies for solving German language capitalization tasks. The four strategies (syntactic, semantic, morphological and pragmatic) do partially overlap. We have shown that the different learning strategy adoption rates can be observed in user error profiles (RQ1). This opens up opportunities for individualized training and therefore for higher motivation and learning success for students.

Second, we found that learners prefer different strategies depending on their grade level and gender (RQ2). This information can be used to adapt the online platform orthografietrainer.net to various learner levels. For example, based on this information, the difficulty of the words can be calculated more specifically for each user, and task sequencing can be adjusted to be neither too difficult nor too easy. This reduces the potential for frustration caused by tasks that are too difficult and also increases motivation. Furthermore, with tasks that represent typical sources of error for a user, the platform could display appropriate tips and hints. If the error analysis results are made available to the teacher on the dashboard of the online platform, he or she can see which rules have not yet been observed by the students and can adapt lessons accordingly. Further research could include the implementation and subsequent validation through A/B testing of such improvements.

Finally, our findings lead to a better understanding of how capitalization is learned and taught (RQ3). Our research shows that there is a great discrepancy between which strategies are taught in class and which strategies are used by students. We therefore suggest that highly granular learning process data can not only provide insights into learners' abilities and enable individualized learning content but also inform curriculum development.

Other future analyses could investigate whether the learning strategies can be applied to other grammatical areas, such as separated and combined spelling.

7. ACKNOWLEDGEMENTS

This research was funded by the Federal Ministry of Education and Research of Germany in the framework "Digitalisierung im Bildungsbereich" (project number 01JD1812A).

REFERENCES

[1] Abu-rabia, S. 2002. Reading in a root–based–morphology language: the case of Arabic. *Journal of Research in Reading* 25, 3, 299–309. DOI= https://doi.org/10.1111/1467-9817.00177.

[2] Abu-rabia, S. and Taha, H. 2004. Reading and spelling error analysis of native Arabic dyslexic readers. *Reading and Writing: An Interdisciplinary Journal* 17, 7-8, 651–690. DOI= https://doi.org/10.1007/s11145-004-2657-x.

[3] Alhaisoni, E. M., Al-Zuoud, K. M., and Gaudel, D. R. 2015. Analysis of Spelling Errors of Saudi Beginner Learners of English Enrolled in an Intensive English Language Program. *English Language Teaching* 8, 3, 185–192.

[4] Al-Oudat, A. 2017. Spelling Errors in English Writing Committed by English-Major Students at BAU. *Journal of Literature, Languages and Linguistics* 32.

[5] Bredel, U., Müller, A., and Hinney, G. 2010. Schriftsystem und Schrifterwerb. Linguistisch - didaktisch - empirisch 289.

[6] Brusilovsky, P. and Peylo, C. 2003. Adaptive and Intelligent Web-based Educational Systems. *International Journal of Artificial Intelligence in Education* 13, 159–172.

[7] Caravolas, M. and Volín, J. 2001. Phonological spelling errors among dyslexic children learning a transparent orthography: the case of Czech. *Dyslexia* 7, 4, 229–245. DOI= https://doi.org/10.1002/dys.206.

[8] Chen, W., Joe-Wong, C., Brinton, C. G., Zheng, L., and Cao, D. 2018. Principles for Assessing Adaptive Online Courses. *International Educational Data Mining Society*.

[9] Cook, V. 1999. Teaching spelling.

[10] Defior, S. and Serrano, F. 2005. The initial development of spelling in Spanish: From global to analytical. *Reading and Writing* 18, 1, 81–98. DOI= https://doi.org/10.1007/s11145-004-5893-1.

[11] Fang, Y., Shubeck, K., Lippert, A., Chen, Q., Shi, G., Feng, S., Gatewood, J., Chen, S., Cai, Z., Pavlik, P., Frijters, J., Greenberg, D., and Graesser, A. 2018. Clustering the Learning Patterns of Adults with Low Literacy Skills Interacting with an Intelligent Tutoring System. *Proceedings of the 11th International Conference on Educational Data Mining* 11.

[12] Gaudioso, E. and Boticario, J. G. 2003. Towards web-based adaptive learning communities. *Proceedings of the 11th International Conference on Artificial Intelligence in Education* 11.

[13] Klašnja-Milićević, A., Vesin, B., Ivanović, M., and Budimac, Z. 2011. E-Learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & Education* 56, 3, 885–899. DOI= https://doi.org/10.1016/j.compedu.2010.11.001.

[14] Landerl, K. and Wimmer, H. 2000. Deficits in phoneme segmentation are not the core problem of dyslexia: Evidence from German and English children. *Applied Psycholinguistics* 21, 2, 243–262. DOI= https://doi.org/10.1017/s0142716400002058.

[15] Lange, J. *Über den Rat*. Retrieved April 26, 2021 from https://www.rechtschreibrat.com/ueber-den-rat/.

[16] Maas, U. 1992. *Grundzüge der deutschen Orthographie*. Niemayer, Tübingen.

[17] Muhammad, A., Zhou, Q., Beydoun, G., Xu, D., and Shen, J. 2016. Learning path adaptation in online learning systems. In *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 421–426. DOI= https://doi.org/10.1109/CSCWD.2016.7566026.

[18] Müller, H.-G. 2008. Das Onlineportal Orthografietrainer.net - Lernpsychologische und didaktische Überlegungen zu einem automatisierten digitalen Übungskonzept.

[19] Müller, H.-G. 2014. Zur textpragmatischen Funktion der Groß- und Kleinschreibung des Deutschen. *ZGL Zeitschrift für germanistische Linguistik* 42, 1, 1–25.

[20] Müller, H.-G. 2016. Der Majuskelgebrauch im Deutschen. Groß- und Kleinschreibung theoretisch, empirisch, ontogenetisch. *Reihe Germanistische Linguistik* 305, Habermann, M. and Hausendorf, H., Eds. Berlin: de Gruyter.

[21] Nimz, K. and Möhlmann, H. 2021. Zum Einfluss orthographischer Fehler in Bewerbungsschreiben. Eine experimentelle Untersuchung mit Personalverantwortlichen. In *Neue Wege des Orthografieerwerbs. Forschung-Vermittlung-Reflexion*, Kepser, M., Schallenberger, S. and Müller, H.-G. Eds. Lemberger, Wien.

[22] Premlatha, K. R. and Geetha, T. V. 2015. Learning content design and learner adaptation for adaptive e-learning environment: a survey. *Artificial Intelligence Review* 44, 4, 443–465. DOI= https://doi.org/10.1007/s10462-015-9432-z.

[23] Protopapas, A., Fakou, A., Drakopoulou, S., Skaloumbakas, C., and Mouzaki, A. 2013. What do spelling errors tell us? Classification and analysis of errors made by Greek schoolchildren with and without dyslexia. *Reading and Writing* 26, 5, 615–646. DOI= https://doi.org/10.1007/s11145-012-9378-3.

[24] Röber-Siekmeyer, C. 2003. *Ein anderer Weg zur Groß- und Kleinschreibung*. Klett-Grundschulverlag, Leipzig.

[25] Surjono, H. D. 2015. The Effects of Multimedia and Learning Style on Student Achievement in Online Electronics Course. *Turkish Online Journal of Educational Technology - TOJET* 14, 1, 116–122.

[26] Thomé, G. and Eichler, W. 2008. Rechtschreiben Deutsch. In *Unterricht und Kompetenzerwerb in Deutsch und Englisch. Ergebnisse der DESI-Studie*, Klieme, E. Ed. Beltz Pädagogik. Beltz, Weinheim, Basel, 104–111.

[27] Varnhagen, C. K. 2000. SHOOT THE MESSENGER AND DISREGARD THE MESSAGE? CHILDRENS ATTITUDES TOWARD SPELLING. *Reading Psychology* 21, 2, 115–128. DOI= https://doi.org/10.1080/02702710050084446.

[28] Wahl, S., Rautenberg, I., and Helms, S. 2017. Evaluation einer syntaxbasierten Didaktik zur satzinternen Großschreibung. *Didaktik Deutsch: Halbjahresschrift für die Didaktik der deutschen Sprache und Literatur* 22, 42, 32–52.

Integrating Deep Learning into An Automated Feedback Generation System for Automated Essay Scoring

Chang Lu, Maria Cutumisu
Department of Educational Psychology
Faculty of Education, University of Alberta
{clu4, cutumisu}@ualberta.ca

ABSTRACT

Digitalization and automation of test administration, score reporting, and feedback provision have the potential to benefit large-scale and formative assessments. Many studies on automated essay scoring (AES) and feedback generation systems were published in the last decade, but few connected AES and feedback generation within a unified framework. Recent advancements in machine learning algorithms enable researchers to develop more models that explore the potential of automated assessments in education. This study makes the following contributions. First, it implements, compares, and contrasts three AES algorithms with word-embedding and deep learning models (CNN, LSTM, and Bi-LSTM). Second, it proposes a novel automated feedback generation algorithm based on the Constrained Metropolis-Hastings Sampling (CGMH). Third, it builds a classifier to integrate AES and feedback generation into a systematic framework. Results show that (1) the scoring accuracy of the AES algorithm outperforms that of state-of-the-art models; and (2) the CGMH method generates semantically-related feedback sentences. The findings support the feasibility of an automated system that combines essay scoring with feedback generation. Implications may lead to the development of models that reveal linguistic features, while achieving high scoring accuracy, as well as to the creation of feedback corpora to generate more semantically-related and sentiment-appropriate feedback.

Keywords

Automated essay scoring, deep learning, feedback generation, assessment, machine learning, natural language processing

1. INTRODUCTION

Automatic essay scoring (AES), the task of machine-grading essays or constructed-response items, has been gaining attention due to technology-powered advances in educational assessment [16]. The goal of AES is to produce reliable and valid scores using machine scoring rather than human scoring [43]. Previous research has made advances in automatic grading essays with handcrafted features [16, 30, 40]. Currently, with the availability of large volumes of trainable corpora extracted online and the development of models for word representation in the Natural Language Processing (NLP), deep learning approaches have produced highly reliable scores using text classification methods [10, 12, 22]. However, few studies have approached automated essay scoring and automated feedback generation to achieve a fully automated computer-based testing system (CBT).

Earlier attempts at implementing feedback have been made using real-time online tutoring by humans [18, 31]. Findings show that human tutoring is effective at improving students' performance, but it is time consuming and labor intensive. Also, human tutoring is not applicable to large-scale practice and open-ended platforms with large numbers of students. Research on automated feedback generation emerged in the last decade to fill this gap by developing tools to scaffold students within computer-based testing environments [24, 36]. Previous studies have focused on generating formative feedback using rule-based approaches [3, 38]. Although rule-based feedback generation is relatively easy to achieve and the generated sentences can be considered to be appropriate feedback, this approach is usually restricted to pre-designed templates. Recent efforts have been made to engage students in more communicative and adaptive environments and to propose feedback-generation frameworks using sentence generation with constraints, where the constraints are often defined by domain-specific terms [8, 11]. Nevertheless, few studies have empirically examined automated language generation in CBTs. This study proposes a framework that introduces an algorithm based on deep learning models with an unsupervised sentence-generation approach to automatically grade essays and to generate feedback.

2. RELATED WORK

2.1 Automated Essay Scoring

Automated essay scoring constitutes the task of automatically assigning scores to written essays based on features or characteristics in the text. Several systems for Automated Essay Scoring (AES) have already been developed and used in large-scale high-stakes assessments for several decades. Page [33] designed the first intelligent scoring system, Project Essay Grade (PEG), using simple linear regression with hand-crafted features such as essay length and proposition counts to perform text classification tasks based on these features. Since then, other systems for automated essay scoring emerged such as Intelligent Essay Assessor [25], e-rater [6], IntelliMetric [41], and My Access! [41]. Several AES methods have been later adopted to make predictions on student writing scores. Yannakoudakis et al. [44] approached AES as a preference-ranking problem and evaluated essays based on pairwise comparisons of features, such as POS n-grams features and complex grammatical features. Gierl et al. [16] demonstrated the application of AES in medical exams with Support Vector Machine (SVM). Phandi et al. [34] approaches AES with Bayesian Linear Ridge Regression. Taghipour and Ng [40] designed an 'Enhanced AI Scoring Engine' (EASE) based on four genres of

Chang Lu and Maria Cutumisu "Integrating Deep Learning into An Automated Feedback Generation System for Automated Essay Scoring". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 573-579. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

features: length-based features, Parts-of-Speech (POS), word-prompt overlap, and bag of n-grams. These features were fed into several model architectures such as Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN)-variants, namely Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM), to compare the prediction performance of the models. Phandi et al.'s [34] attempt to employ deep learning models to predict essay scores was later used as a baseline for related studies. Previous studies on automated essay scoring rely heavily on hand-crafted feature engineering and knowledge on linguistic discourse. Inspired by recent advances of deep learning models and word embedding techniques, a substantial body of literature has emerged, which has contributed to applying deep learning methods in automated essay scoring tasks. Alikaniotis et al. [1] implemented a two-layer bidirectional LSTM with score-specific word embeddings to learn essay representations and conduct AES tasks. The proposed model outperformed the baseline SVM model. Later, Dong and Zhang [9] established a three-layer model architecture combining CNN for character representation and LSTM for sentence representation with an extra attention-pooling layer, which performed better than Taghipour and Ng's [40] model and their two-layer CNN model. Taghipour and Ng's attempt of combining feature engineering and deep learning models inspired later trials of applying word embeddings and deep learning methods on AES.

2.2 Automated Feedback Generation

Providing feedback is a key ingredient in performance improvement. In education, feedback is defined as the information provided by an agent regarding aspects of one's performance or understanding [17]. High-quality personalized and timely feedback can improve learners' performance [17], but feedback provision is often reported as the long-standing weakness of ITSs and computer-based assessment systems [27]. On the one hand, students complain that they receive too little quality feedback in the process of learning [5, 13]. On the other hand, students are reported to misuse and abuse the feedback or hints provided by the ITSs [37]. Thus, knowing how and when to provide real-time personalized feedback that guides and motivates students' learning remains a challenge.

Williams and Dreher [42] advocated the potential of fully-automated systems that perform both scoring and feedback provision with machines in tasks such as essay grading. Previous efforts have been made to produce feedback in intelligent tutoring or assessment systems [7, 19, 21] for various disciplines, such as computer science [14, 23], information and communication technology (ICT [7]), and English as a Second Language (ESL [26]). However, most automated assessment systems adopt a template-based method to generate feedback [4, 26, 42], which usually produces feedback that is limited to fixed expressions.

Recent advances on constrained sentence generation shed some light onto flexible feedback generation. For example, Su et al. [39] proposes a Gibbs sampling method to meet the constraints of sentiment control. However, Gibbs sampling is not able to vary the sentence length or handle keywords when generating the sentence. Miao et al. [32] extends the Gibbs sampling to a novel unsupervised sampling approach, named Constrained Generation by Metropolis-Hastings sampling (CGMH). The CGMH is a subtype of the Markov Chain Monte Carlo (MCMC [15]) methods. The CGMH allows for more flexible operations on word tokens in a sentence space, thus it is easier to generate content with constraints and varying sentence lengths. Miao et al. [32] tested the CGMH on three tasks including key-to-sentence generation with hard constraints, paraphrase, and error correction with soft constraints. The CGMH method outperformed state-of-art sentence-generation algorithms. Yet, one of the reasons that the research on automated feedback generation with NLP is lagging behind may be that there is no publicly-available feedback corpus.

2.3 Present Study

We propose an AES and an automated feedback generation framework to support students' performance. Specifically, the current study implements three deep learning models for automated essay scoring: (1) CNN; (2) CNN and LSTM; and (3) CNN and Bi-LSTM. In addition, a novel unsupervised sentence-generation approach uses CGMH to automatically provide feedback for test takers based on their predicted essay scores. The remaining sections are guided by the following research questions:

1. To what extent can the AES algorithms generate accurate performance on essay scoring?

2. To what extent can the CGMH algorithms generate fluent and semantically-related feedback?

The contributions of the present study are three-fold. First, the study advances computer-based testing by incorporating automated feedback generation into the assessment framework, especially for unstructured text (e.g., essays). Second, the flexible unsupervised learning approach creates a corpus of semantically-related and sentiment-appropriate feedback for scaffolding. Third, the scalable automatic assessment and feedback provision system is automated and performs accurately, which paves the way for future implementations of feedback generation for various domains within intelligent tutoring systems.

3. METHOD

3.1 Datasets and Corpus

The dataset for the AES task was retrieved from a Kaggle challenge named Automated Student Assessment Prize (ASAP) sponsored by the Hewlett Foundation in 2012 and detailed in Table 1.

Table 1. Summary of ASAP dataset

Prompt	Genre	Grade Level	Training set size	Score Range	Ave Length
1	persuasive /narrative / expository	8	1783	2-12	350
2	persuasive /narrative / expository	10	1800	1-6	350
3	source dependent	10	1726	0-3	350
4	source dependent	10	1772	0-3	150
5	source dependent	8	1805	0-4	150
6	source dependent	10	1800	0-4	150
7	persuasive /narrative/ expository	7	1569	0-30	250
8	persuasive /narrative / expository	10	723	0-60	650

The ASAP is the benchmark dataset for piloting AES studies. It consists of 8 prompts and 4 genres, including persuasive, narrative,

expository, and source-dependent responses. In total, 12,979 essays were released. Since the ASAP has not made the official test sets publicly available, we used 60% of the training set for training, 20% for validation, and 20% for testing. We first performed text cleaning, tokenization, and padding. Then, we used Stanford's publicly-available GloVe 300-dimensional model to conduct word embeddings [35]. The GloVe 300-dimensional embeddings were trained on 6 billion words scraped from Wikipedia and other web texts. The writing prompts have different score ranges as shown in Table 1. To address the issue of inconsistent score ranges, we followed Phandi et al.'s [34], Taghipour and Ng's [40], and Dong and Zhang's [9] method by approaching the AES as a regression task, rescaling the essay score to [0, 1] in the training, validating, and test stages, and projecting the scores back to their original scales in the evaluation stage.

The corpus used to train language models for sentence generation consisted of the publicly-available IMDB dataset, which contains 25,000 positive reviews and 25,000 negative reviews. The dataset was split into three parts: the training set consisted of 20,000 negative reviews and 20,000 positive reviews, the validation test consisted of 1,250 negative and 1,250 positive reviews, and the test set consisted of 1,250 negative and 1,250 positive reviews. A third-party corpus, the Reuters corpus from NLTK, was used for evaluation of the quality of the generated sentences.

3.1.1 AES Step

The present study was conducted in two steps. Step 1 addressed automated essay scoring, whereas Step 2 addressed automated feedback generation. An essay performance classifier was added to synthesize the two steps into a unified framework.

In the AES task, three deep-learning algorithms were implemented and compared regarding their performance and efficiency to select the optimal algorithm as the foundation of the feedback generation step: CNN, CNN + LSTM, and CNN + Bi-LSTM. The convolutional layer is seen as a function that could learn features from n-grams, and can be represented as:

$$Z_i = f\left(W_z\left[x_i^j : x_i^{j+h_w-1}\right] + b_z\right),$$

where x_i is the ith embedded word, W_z is the weight matrix, b_z is the bias vector, h_w is the window size of the convolutional layer, f is a non-linear activation function (i.e., sigmoid, tanh, or ReLu), and Z_i is the output of feature representation.

LSTM is an RNN model for processing sequence data [20]. The unit or memory cell of LSTM consists of an input gate, a forget gate, and an output gate to control information flow. The gates decide preserving, forgetting, and passing information as a vector sequence at each time step.

More specifically, assuming there are *T* sentences in an essay in total, the composite functions at sentence *t* can be written as:

$$i_t = \sigma(W_i s_t + U_i h_{t-1} + b_i),$$
$$f_t = \sigma(W_f s_t + U_f h_{t-1} + b_f),$$
$$g_t = tanh(W_g s_t + U_g h_{t-1} + b_g),$$
$$c_t = i_t \odot g_t + f_t c_{t-1},$$
$$O_t = \sigma(W_o s_t + U_o h_{t-1} + b_o),$$
$$h_t = O_t \odot \tanh(c_t),$$

in which s_t is the input vector, h_t is the output vector, W_i, W_f, W_g, W_o, U_i, U_f, U_g, U_o are the estimated weight matrices, and b_i, b_f, b_g, b_o are the bias vectors.

Bi-LSTM is an extension of unidirectional-LSTM for deeper representations. Compared with unidirectional-LSTM that can only preserve and pass information from history, Bi-LSTM can also make use of information from future. In AES tasks, Bi-LSTM could process the words in the input vector in both a forward and a backward manner. The composite function for Bi-LSTM is similar with LSTM:

$$y_t = W_{yh}\begin{pmatrix}\overrightarrow{h_t} \\ \overleftarrow{h_t}\end{pmatrix} + b_y.$$

The summary of the model architectures for the three models is shown in Table 2.

Table 2. Model architecture summary

Layer	Hyperparameter	Value
CNN		
Embeddings	dimension	300
Convolutional	filters, kernel size	100, 5
CNN + LSTM		
Embeddings	dimension	300
Convolutional	filters, kernel size	100, 5
LSTM	units	32
CNN + Bi-LSTM		
Embeddings	dimension	300
Convolutional	filters, kernel size	100, 5
Bi-LSTM	layers	16

3.1.2 Feedback Generation Step

The feedback generation phase included two steps. In Step 1 (Corpus Development), we will develop a corpus of feedback using CGMH based on the expert-derived essay descriptors. In Step 2 (Feedback Generation & Provision), we will develop feedback based on the essay scores provided by the AES algorithms.

Table 3 shows the part of the essay-scoring rubrics (the score ranged from 1 to 6) and descriptors developed by experts.

Table 3. Sample descriptor for Essay Prompt 1

Score	Descriptors
1	An undeveloped response that may take a position but offers no more than very minimal support.
Element	Contains few or vague details.
	Is awkward and fragmented.
	May be difficult to read and understand.
	May show no awareness of audience.
2	An under-developed response that may or may not take a position.
Element	Contains only general reasons with unelaborated and/or list-like details.
	Shows little or no evidence of organization.

	May be awkward and confused or simplistic.
	May show little awareness of audience.

To expand the corpus, we will adopt the *Constrained Sentence Generation by Metropolis-Hastings Sampling* method (CGMH [32]) to perform unsupervised paraphrase generation. The CGMH facilitates the generation of content with constraints and varying sentence lengths. Miao et al. [32] tested the CGMH on three tasks, including keywords-to-sentence generation with hard constraints, paraphrase, and error correction with soft constraints. In the present research, we will implement unsupervised paraphrasing to augment the feedback corpus. Specifically, we will first train a language model based on the IMDB review corpus [29]. The IMDB dataset consists of 25,000 positive and 25,000 negative movie reviews. It was selected for the feedback-generation task for the following reasons. First, to date, there is no database of academic feedback, the IMDB was the closest commentary corpus available. More importantly, this corpus is split into positive and negative phrases, which makes it domain-independent. Thus, it can transfer more easily to other domains. Then, we will perform the paraphrase generation.

A Markov model is used to train the language model on the selected corpus. The Markov Chain is commonly used to model natural language as a function of the probability that a word appearing in position *n* is only dependent on the previous *z* ϵ [1, n-1] such that:

$$p(w_1, w_2, \dots, w_n) = p(w_1)\, p(w_2|w_1), \dots\dots, p(w_n|w_{n-z}, \dots, w_{n-1}),$$

where $p(w_1, w_2, \dots, w_n)$ refers to the probability of a specific sentence based on the trained corpus, that is, the joint probability of all words within the sentence. In the present research, we used forward-backward dynamic programming to train the language model.

In Step 2 (feedback paraphrase), we performed the CGMH task of unsupervised sentence paraphrasing. The CGMH is concerned with a goal of stationary distribution that defines the sentence distribution sampled from the corpus and three actions, namely, replacement, insertion, and deletion. Specifically, $\pi(x)$ was set as the distribution from which we plan to sample sentences, where *x* denotes a particular sentence and x_0 refers to the feedback template that is fed to the algorithm at time step 0. The MH sampler either accepts or rejects a word from the given distributions $\pi(x)$ to finally form a desired joint distribution of all words based on a predefined stationary distribution. The process is intuitive, as it mainly involves two actions: accepting or rejecting a word monitored by the acceptance rate α:

$$\alpha = \min\{1, \frac{\pi(x')g(x_{t-1}|x')}{\pi(x_{t-1})g(x'|x_{t-1})}\}$$

At time step *t*, the word sampling is conducted to update the previous state *x* to a candidate distribution x' from a proposed distribution $g(x'|x_{t-1})$, where x_{t-1} refers to the distribution from previous step (*t-1*), thus $x' = x_t$. Therefore, α determines the acceptance or rejection of a sample. In our paraphrase generation, the desired distribution denotes the most likely and logical sentence related to the original sentence.

At each step, a selected word in the sentence will be updated by the actions such as insertion, deletion, and replacement, randomly, where the respective probabilities are $[p_{insert}, p_{delete}, p_{replace}]$. At the first time step, these probabilities are set as being equal. At the following step, if *Replacement* is applied on a selected word w_m in a sentence $x = [w_1, w_2, \dots, w_{m-1}, w_m, w_{m+1}, \dots, w_n]$, then the conditional probability of choosing w_m^{new} to replace w_m to form candidate sentence x' from *x* can be computed as:

$$g_{replace}(x'|x) = \pi(w_m^{new}|x_{-m}) = \frac{\pi(w_1, w_2, \dots, w_{m-1}, w_m^{new}, w_{m+1}, \dots, w_n)}{\sum_{w \in V}(w_1, w_2, \dots, w_{m-1}, w, w_{m+1}, \dots, w_n)},$$

where *V* refers to the vocabulary, and w_m is the selected word. If, on the other hand, *Insertion* is applied, an additional step of inserting a placeholder will be conducted before taking the action *Replacement*, and then a real word will be sampled to replace the placeholder token with the *Replacement* token. Finally, if *Deletion* is applied, the w_m word selected will be deleted, and $g_{deletion}(x'|x)$ =1 if $x' = [w_1, w_2, \dots, w_{m-1}, w_{m+1}, \dots, w_n]$, and 0 otherwise. The detailed settings of the sentence-generation phase, including the hyperparameter values determined in the tuning process are included in Table 4.

Table 4. MCMC hyperparameter

Hyperparameters	**Value**
Dictionary size	50,000
Hidden nodes per LSTM layer	300
Number of steps	50
Maximum sentence length	50
Max epoch	30
Minimum of Sentence Length	7
Initial action probability	[0.3, 0.3, 0.3, 0.1]

3.1.3 Synthesis

One important purpose of the present study is to develop a framework linking automated essay scoring and automated feedback generation. Thus, the study can be decomposed in two parts: a supervised text classification task using CNN and RNN models and an unsupervised learning paraphrase generation task using MCMC sampling method with constraints. In the synthesis, a performance classifier was applied to extract feedback that corresponds to the score that is assigned by the AES algorithms.

3.2 Evaluation Metrics

The objective of the AES training stage is to minimize the mean squared error (MSE) between the scores provided by human raters and the prediction scores generated by the models.

In the automated essay scoring tasks, several measures including the quadratic weighted kappa (QWK [9, 10]), exact agreement, and alternate-form reliabilities [2] have been used to evaluate the performance of AES models in previous studies. In the current study, we present the results of QWK, which measures the degree of agreement between human raters and the machine on one essay and can be calculated by:

$$\text{QWK} = 1 - \frac{\sum_{i,j} W_{i,j}\, O_{i,j}}{\sum_{i,j} W_{i,j}\, E_{i,j}}.$$

where $W_{i,j}$ is calculated by $W_{i,j} = \frac{(i-j)^2}{(N-1)^2}$ (i: represents the human-rated score; j: represents machine-rated score; N: represents the score range), $O_{i,j}$ represents the number of essays that receive a rating i by the human and a rating j by the machine, and E is the outer product of the histogram vectors of the two scores. According to Williamson, Xi, and Breyer [43], QWK scores higher than 0.7 indicate high accuracy.

For the feedback generation task, we used several measures to evaluate the generated sentences. The first step is concerned with language model training, whereas the second step is concerned with generating sentences with the MCMC sampling method. More specifically, we first reported the training process of the language model over epochs. The objective of the first training process is to minimize the perplexity of the language model, which can be calculated by:

$$PPL = 2^{-\frac{1}{N}\sum_{i=1}^{N} \log\, p(w_i)},$$

where N equals the number of words in the corpus and $p(w_i)$ indicates the probability of a word appearing in the position. The lower the PPL is, the more precisely the corpus is modeled.

For the generated sentences, we evaluated the model performance using two measures. First, we computed the Negative Likelihood (NLL) of the sentences to evaluate their fluency using the Reuters corpus released by NLTK modules. The lower the NLL is, the more fluent the sentences are. Second, we invited two volunteers to rate the quality of 50 pieces of feedback in terms of the sentence fluency and relatedness at a scale of 0-1, and the higher the scores are, the more fluent and related the generated feedback sentences are.

4. RESULTS

4.1 Rating Accuracy of AES Algorithms

The results show that, for Prompt 1, the most accurate algorithm is CNN + Bi-LSTM, whereas for Prompts 2 to 8, the most accurate algorithm is CNN + LSTM. The average QWK of CNN + LSTM reaches 0.734, as shown in Table 5. In general, the models that integrate LSTM/Bi-LSTM perform better than CNN. Compared with the baseline [34], the CNN+LSTM model in the present study performs better on Writing Prompt 1-7, but poorer on Prompt 8. In addition, the average QWK of CNN+LSTM also outperforms the baseline model [34].

Table 5. Comparisons of QWK of the implemented models

Prompt	CNN	CNN + LSTM	CNN + Bi-LSTM	Phandi et al., 2015
1	0.81	0.87	0.88	0.76
2.1	0.62	0.64	0.52	0.61
2.2	0.51	0.61	0.51	-*
3	0.73	0.63	0.62	0.62
4	0.83	0.83	0.72	0.74
5	0.77	0.86	0.76	0.78
6	0.77	0.85	0.8	0.78
7	0.72	0.79	0.75	0.73
8	0.35	0.53	0.54	0.62
Ave	0.68	0.73	0.68	0.71

Note: *indicates that prompts 2.1 and 2.2 were combined into a single score.

Results of the average QWK across genres (e.g., persuasive, narrative, and expository) and source-independent writing can be found in Table 6, which shows that CNN+LSTM outperformed CNN and CNN+Bi-LSTM on both genres. However, the three models all performed poorly on the persuasive, narrative, and expository criteria. The results are consistent with previous studies [34], as the models generally have better predictions on the prompts with smaller score ranges. The wide-score range may cause more complexities for the training process of deep learning models. In addition, previous studies on applying deep neural networks in AES yielded similar results showing that models generally performed poorly on Prompts 2 and 8 [9, 10, 34, 40]. The present study also found that the three deep learning algorithms showed higher efficiency on scoring certain types of genres of writing, but less accuracy on Prompts 2, 3, and 8. One possible explanation is that, for Prompt 2, two domain scores instead of one single global score are provided. The inherent inconsistency or low reliability of a single human rater's scoring makes it difficult for machines to learn the scoring pattern. While for Prompt 8, the score range is 0 to 60, as shown in Table 1. Compared with other prompts whose score ranges are narrow (0 to 3 or 0 to 4), this extremely wide range (i.e., the categories of the outcome variable) may hinder the learning process of deep learning models.

Table 6. Average QWK across genres

QWK	persuasive /narrative/ expository (Prompt 1,2,7,8)	source independent (Prompt 3,4,5,6)
CNN	0.601	0.775
CNN+LSTM	0.688	0.793
CNN+Bi-LSTM	0.640	0.726

4.2 Runtime of AES

Prediction accuracy is of utmost priority in machine learning. However, in a fully-automated scoring and reporting system, scoring efficiency represented as the time it took to run one epoch (i.e., the runtime) also plays an important role. Table 7 shows the average runtime for one epoch of the three models: CNN was the fastest of the three on average. Therefore, it can be concluded that CNN+LSTM has the highest performance, but also has relatively high efficiency (i.e., it is the second fastest algorithm of the three). Thus, it was chosen as the AES algorithm for the feedback-generation step.

Table 7. Average runtime and memory

Model	Runtime for one epoch	N of Parameters
CNN	51s	5117233
CNN + LSTM	53s	4988977
CNN + Bi-LSTM	55s	4986929

4.3 NLL of Generated Feedback and Human Ratings

For the sentence generation process, the generated sentences were the ones with the lowest NLL after 50 steps of running. The feedback phrases were generated using a sentence paraphrasing CGMH approach before being passed on to the performance classifier. The feedback templates were sampled from the ASAP rating descriptors and feedback phrases were generated based on the language model trained on the IMDB dataset.

Figure 1 presents the training process of the language model, and Table 8 shows the NLL and human-rater evaluations of the generated sentences regarding *Fluency* and *Relatedness* on a scale of 0 to 1. The higher the scores, the more fluent and related the generated feedback sentences. The results revealed that the MCMC method is able to generate fluent and semantically-relevant sentences.

Table 8. NLL and rater evaluation on sentence generation

Evaluation Methods	Measures
NLL	10.01
Human Rating: Fluency	0.62
Human Rating: Relatedness	0.52

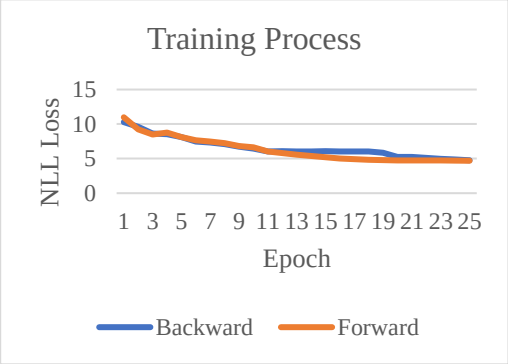


Figure 1. Learning curves of the training process of the language model (NLL convergence w.r.t. epochs).

5. DISCUSSION AND CONCLUSION

This study proposed and implemented a novel framework for an automated assessment and reporting framework with a combination of supervised deep learning models and unsupervised MCMC sampling method. Specifically, this study compared the performances of three models, namely CNN, CNN+LSTM, and CNN+Bi-LSTM, on AES tasks in the same context. Results revealed that CNN+LSTM demonstrated the highest performance on the AES tasks among the three algorithms. Moreover, the CNN+LSTM outperformed the baseline model on seven out of eight writing prompts, which demonstrates the potential of word-embeddings and deep learning models on automated essay scoring.

A recent literature review revealed that text-based feedback was more effective in improving performance [28]. Providing feedback within digital learning and assessment systems is essential for students' self-directed learning. However, it is laborious to manually devise a large amount of expert-derived quality feedback. Compared with sentence-generation supervised-learning methods, the CGMH sentence-paraphrasing unsupervised-learning method can augment the expert-driven feedback template corpus by generating feedback phrases with higher efficiency and flexibility. Thus, the proposed method is promising in promoting text-based feedback generation within automated assessment systems. Results of the current study could facilitate future implementations and validations of personalized automated feedback provision for ITSs and other virtual learning systems.

6. LIMITATIONS AND FUTURE WORK

We identified several limitations in the present study. First, this study does not empirically validate the AES and the automated feedback generation system in educational settings. Future research will be conducted to provide empirical evidence on the validity and efficiency of the framework. Second, the present framework generates feedback using a holistic score for essays. Future research will incorporate linguistic components into AES to enhance the interpretability of the scoring results and to generate more fine-grained feedback.

7. ACKNOWLEDGEMENT

We would like to thank the reviewers for their helpful feedback and the following granting agencies for supporting this research: the Social Sciences and Humanities Research Council of Canada - Insight Development Grant (SSHRC IDG) RES0034954 and Insight Grant (SSHRC IG) RES0048110, Natural Sciences and Engineering Research Council Discovery Grant (NSERC DG) RES0043209, Killam Cornerstone Operating Grant RES0043207, Alberta Innovates, and Alberta Advanced Education.

8. REFERENCES

[1] D. Alikaniotis, H. Yannakoudakis, and M. Rei. Automatic text scoring using neural networks. *arXiv:1606.04289*, 2016.

[2] Y. Attali and J. Burstein. Automated essay scoring with e-rater® V. 2. *The Journal of Technology, Learning and Assessment*, 4(3):i-21, 2006.

[3] T. Barnes and J. Stamper. Automatic hint generation for logic proof tutoring using historical data. *Journal of Educational Technology & Society*, 13(1):3-12, 2010.

[4] P. Blayney and M. Freeman. Automated formative feedback and summative assessment using individualised spreadsheet assignments. *Australasian Journal of Educational Technology*, 20(2):209-231, 2004.

[5] D. Boud and E. Molloy. (Eds.). *Feedback in Higher and Professional Education: Understanding it and Doing it Well*. Routledge, 2013.

[6] J. Burstein, J. Tetreault, and N. Madnani. The E-rater® automated essay scoring system. In *Handbook of Automated Essay Evaluation*, pages 77-89. Routledge, 2013.

[7] J. Debuse, M. Lawley, and R. Shibl. The implementation of an automated assessment feedback and quality assurance system for ICT courses. *Journal of Information Systems Education*, 18(4):491-502, 2007.

[8] B. Di Eugenio, D. Fossati, D. Yu, S. M. Haller, and M. Glass. Natural Language Generation for Intelligent Tutoring Systems: A case study. In *AIED*, pages 217-224, 2005.

[9] F. Dong and Y. Zhang. Automatic features for essay scoring–an empirical study. In *Proceedings of the 2016 Conference on EMNLP*, pages 1072-1077, 2016.

[10] F. Dong, Y. Zhang and J. Yang. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on* CoNLL, pages 153-162, 2017.

[11] M. Dzikovska, N. Steinhauser, E. Farrow, J. Moore, and G. Campbell, G. BEETLE II: Deep natural language understanding and automatic feedback generation for intelligent tutoring in basic electricity and electronics. *International Journal of Artificial Intelligence in Education*, 24(3):284-332, 2014.

[12] Y. Farag, H. Yannakoudakis, and T. Briscoe. Neural automated essay scoring and coherence modeling for adversarially crafted input. *arXiv:1804.06898*, 2018.

[13] P. Ferguson. Student perceptions of quality feedback in teacher education. *Assessment & Evaluation in Higher Education*, 36(1):51-62, 2011.

[14] J. Gao, B. Pang, and S. S. Lumetta. Automated feedback framework for introductory programming courses. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 53-58. ACM, 2016.

[15] C. J. Geyer. Practical Markov Chain Monte Carlo. In *Statistical Science*, pages 473-483, 1992.

[16] M. J. Gierl, S. Latifi, H. Lai, A. P. Boulais, and A. De Champlain. Automated essay scoring and the future of educational assessment in medical education. *Medical Education*, 48(10):950-962, 2014.

[17] J. Hattie and H. Timperley. The Power of Feedback. Review of Educational Research. *Review of Educational Research*. 77(1):81-112, 2007.

[18] N. T. Heffernan and K. R. Koedinger. An intelligent tutoring system incorporating a model of an experienced human tutor. In *International Conference on Intelligent Tutoring Systems*, pages 596-608. Springer, 2002.

[19] R. Higgins, P. Hartley, and A. Skelton. The conscientious consumer: Reconsidering the role of assessment feedback in student learning. *Studies in Higher Education*, 27(1):53-64, 2002.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735-1780, 1997.

[21] L. E. Holmes and L. J. Smith. Student evaluations of faculty grading methods. *Journal of Education for Business*, 78(6):318-323, 2003.

[22] C. Jin, B. He, K. Hui, and L. Sun. TDNN: a two-stage deep neural network for prompt-independent automated essay scoring. In *Proceedings of the 56th Annual Meeting of the ACL* (Volume 1: Long Papers), pages 1088-1097, 2018.

[23] H. Keuning, J. Jeuring, and B. Heeren. Towards a systematic review of automated feedback generation for programming exercises. In *Proceedings of the 2016 ACM Conference on ITiCSE*, pages 41-46. ACM, 2016.

[24] E. Kosba, V. Dimitrova, and R. Boyle. Adaptive feedback generation to support teachers in web-based distance education. *User Modeling and User-Adapted Interaction*, 17(4):379-413, 2007.

[25] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259-284, 1998.

[26] M. Liu, Y. Li, W. Xu, and L. Liu. Automated essay feedback generation and its impact on revision. *IEEE Transactions on Learning Technologies*, 10(4):502-513, 2016.

[27] M. Maniktala, C. Cody, A. Isvik, N. Lytle, M. Chi, and T. Barnes. Extending the hint factory for the assistance dilemma: A novel, data-driven HelpNeed predictor for proactive problem-solving help. *Journal of Educational Data Mining*, 12(4): 24-65, 2020.

[28] S. Marwan, N. Lytle, J. J. Williams, and T. Price. The impact of adding textual explanations to next-step hints in a novice programming environment. In *Proceedings of the 2019 ACM Conference on ITiCSE*, pages 520-526, 2019.

[29] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the ACL: HLT*, pages 142-150, 2011.

[30] D. S. McNamara, S. A. Crossley, R. D. Roscoe, L. K. Allen, and J. Dai. A hierarchical classification approach to automated essay scoring. *Assessing Writing*, 23:35-59, 2015.

[31] D. C. Merrill, B. J. Reiser, M. Ranney, and J. G. Trafton. Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *The Journal of the Learning Sciences*, 2(3):277-305, 1992.

[32] N. Miao, H. Zhou, L. Mou, R. Yan, and L. Li. CGMH: Constrained sentence generation by Metropolis-Hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33), pages 6834-6842, 2019.

[33] E. B. Page. Computer grading of student prose, using modern concepts and software. *The Journal of Experimental Education*, 62(2):127-142, 1994.

[34] P. Phandi, K. M. A. Chai, and H. T. Ng. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on EMNLP*, pages 431-439, 2015.

[35] J. Pennington, R. Socher, and C. D. Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the Conference on EMNLP*, pages 1532-1543, 2014.

[36] I. Perikos, F. Grivokostopoulou, and I. Hatzilygeroudis. Assistance and feedback mechanism in an intelligent tutoring system for teaching conversion of natural language into logic. *International Journal of Artificial Intelligence in Education*, 27(3):475-514, 2017.

[37] T. W. Price, R. Zhi, and T. Barnes, T. Hint generation under uncertainty: The effect of hint quality on help-seeking behavior. In *International Conference on Artificial Intelligence in Education*, pages 311-322. Springer, 2017.

[38] S. Shatnawi, M. M. Gaber, and M. Cocea. Automatic content related feedback for MOOCs based on course domain ontology. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 27-35. Springer, 2014.

[39] J. Su, J. Xu, X. Qiu, and X. Huang. Incorporating discriminator in sentence generation: A Gibbs sampling method. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1), pages 5496-5503, 2018.

[40] K. Taghipour and H. T. Ng. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on EMNLP*, pages 1882-1891, 2016.

[41] Vantage Learning. Research summary: IntelliMetricTM scoring accuracy across genres and grade levels, 2006.

[42] R. Williams and H. Dreher. Automatically grading essays with markit?. *Issues in Informing Science and Information Technology*, 1:0693-0700, 2004.

[43] D. M. Williamson, X. Xi, and F. J. Breyer. A framework for evaluation and use of automated scoring. *Educational Measurement: Issues and Practice*, 31(1):2-13, 2012.

[44] H. Yannakoudakis, T. Briscoe, and B. Medlock. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the ACL: HLT*, pages 180-189, 2011.

Investigating SMART Models of Self-Regulation and their Impact on Learning

Stephen Hutt[1], Jaclyn Ocumpaugh[1], Juliana Ma. Alexandra L. Andres[1], Nigel Bosch[2], Luc Paquette[2], Gautam Biswas[3], and Ryan S. Baker[1]
[1] University of Pennsylvania
[2] University of Illinois at Urbana-Champaign
[3] Vanderbilt University
hutts@upenn.edu

ABSTRACT

Self-regulated learning (SRL) is a critical 21st-century skill. In this paper, we examine SRL through the lens of the searching, monitoring, assessing, rehearsing, and translating (SMART) schema for learning operations. We use microanalysis to measure SRL behaviors as students interact with a computer-based learning environment, Betty's Brain. We leverage interaction data, survey data, *in situ* student interviews, and supervised machine learning techniques to predict the proportion of time spent on each of the SMART schema facets, developing models with prediction accuracy ranging from *rho* = .19 for translating to *rho* = .66 for assembling. We examine key interactions between variables in our models and discuss the implications for future SRL research. Finally, we show that both ground truth and predicted values can be used to predict future learning in the system. In fact, the inferred models of SRL outperform the ground truth versions, demonstrating both their generalizability and their potential for using these models to improve adaptive scaffolding for students who are still developing SRL skills.

Keywords

Self Regulation, SMART, Self Regulated Learning, Machine Learning, Student Interviews

1. INTRODUCTION

In traditional classrooms, most support for acquiring self-regulated learning (SRL) strategies comes from teachers, who might check in on projects and/or provide advice about next steps [33] in order to keep students focused on their end goals. However, teachers' external regulation alone is insufficient to encourage educational success [24]; the learner must also develop internal regulation schemas. SRL demands may increase when the student is completing a project in a computer-based learning environment that is no longer teacher-led. The software might scaffold learning activities, but identifying the complex behaviors involved with SRL is still not a typical function of most computer-based learning systems.

In most computer-based learning environments, learners must control, manage, plan, and monitor their learning [12], i.e., implement the definitional components of SRL. SRL has consistently been shown to facilitate knowledge acquisition and retention among learners in a structured and systematic way [12]. As such, work has called for a deeper understanding of SRL impacts in online learning [1, 8, 37].

A range of techniques have been used to better understand SRL both in computer-based learning environments (e.g., [1, 5, 12, 34]) and in other contexts (see [17, 27] for meta-analyses). Research in computer-based learning can be split into two groups: supporting SRL and detecting SRL behaviors [46]. Supporting SRL has taken a number of forms, but in general, these approaches typically scaffold students in either their goal-setting, self-evaluation, help-seeking, self-efficacy, or some combination of these [29]. This might be through verbal prompts (e.g. "Take time to read everything,") [7, 22] or more intricate support systems [25], such as progress bars [14], or tools such as notebooks, that better facilitate student reflection [2, 35].

In terms of detecting SRL in computer-based learning environments, Azevedo and colleagues have (using MetaTutor) considered the role that emotion plays in regulation, posing that affect should be considered as we scaffold SRL behaviors [4]. Segedy et al. [36] used interaction data and coherence analysis to measure self-regulation. Learner behaviors were tracked using log files to assess action coherence (i.e., did a student's actions present a coherent strategy relevant to the current tasks), which was shown to predict learning. Winne et al. [45] also leveraged log data in a scalable system that traces student actions, classifying each learning event into SRL categories in order to better understand student cognition, motivation, and metacognition. We build upon this approach in this work.

While interaction data has been successfully used to detect SRL, a number of researchers argue that this data should not be considered in isolation [3, 37, 40]. Instead, we must also consider contextual factors and individual differences not easily inferred from logs. This work combines interaction data with data from targeted in-situ student interviews and student survey data to predict SRL as characterized by the COPES and subsequent SMART models of SRL [42] (discussed in detail below). We examine the impact of SRL on learning, analyzing contextual and student-level factors that may influence SRL behavior and demonstrating the potential of the latent encoding of SRL for identifying students who need further support.

1.1 Related Works

At a high level, SRL is a process in which learners take initiative to identify their learning goals and then adjust their learning strategies, cognitive resources, motivation, and behavior to optimize their learning outcomes [11, 42]. First characterized in

Stephen Hutt, Jaclyn Ocumpaugh, Juliana Ma. Alexandra L. Andres, Nigel Bosch, Luc Paquette, Gautam Biswas and Ryan Baker "Investigating SMART Models of Self-Regulation and their Impact on Learning". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 580-587. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

1989 [47], SRL is now widely acknowledged as an essential skill for learning in the modern knowledge-driven society [23]. In learning technologies specifically, recent work has called for a deeper understanding of SRL and for learning technology that supports the development of SRL strategies [1, 3, 8, 20, 37].

In order to provide insight into how SRL works, researchers have proposed a number of theoretical models (e.g., [30, 47]). Winne & Hadwin's model [43], grounded in information processing theory, characterizes SRL as a series of events that happen over four recursive stages: (1) task definition, (2) goal setting and planning, (3) studying tactics, and (4) metacognitive adaption of studying techniques. Each stage is then characterized by Conditions, Operations, Products, Evaluations, and Standards (COPES). In later work, Winne subcategorized the COPES model further by detailing five kinds of operations—searching, monitoring, assembling, rehearsing, and translating—known as the SMART model [42].

In the context of educational data mining, we can study SRL by measuring these theoretical constructs and studying their relationships to each other and to external measures (such as achievement). SRL constructs can be measured either online (while an activity is happening) or offline (before or after an activity) [34]. Offline assessments typically rely on self-report questionnaires, but student interviews have also been used. These can be implemented either online and offline and can offer advantages over questionnaires that may limit students to pre-defined answers [16, 40].

Trace analysis is perhaps the main approach used (and endorsed [37]) to measure SRL online. Traces (such as log data) capture learning actions along with additional contextual and timing information, providing a detailed window into a learner's processes and behaviors [40]. This data can support microanalytic approaches, as sequences of actions can be aligned with different facets of a self-regulation model [21, 45]. Models that conceptualize SRL in terms of events or student actions (such as the COPES model [43]) lend themselves more to a trace-based analysis [42] than to offline measurement. However, many researchers argue that trace data should be supplemented with additional measurements (e.g., self-reports or think-alouds) when measuring SRL [3, 37, 40].

1.2 Current Study

The current study was conducted within the context of Betty's Brain, a computer-based learning environment for middle school science. We combine multiple data sources (interaction, surveys, and interview data) to analyze SRL patterns through the lens of Winne's COPES and SMART models [42].

We first demonstrate that combining features from different data sources yields the most successful models of the SMART facets. We present a feature analysis to investigate the key interactions in each model. We next examine how the different facets of SRL influence student learning. We consider not only the ground truth calculations of SMART facets but also our predicted models of these facets, showing that the latter better predicts future student outcomes than the original variables.

To our knowledge, this work presents the first exploration of how student interviews, surveys, and interaction data may be used in concert to predict SRL and learning. This approach provides detailed insight into how we may best support students in an environment where external regulation may be harder to provide.

2. DATA

2.1 The Learning Environment

In this project, we used the learning environment Betty's Brain. This system implements a learning-by-teaching model [9], where students teach a virtual agent named "Betty" by creating a causal map of scientific processes (e.g., thermoregulation or climate change). Betty demonstrates her "learning" by taking quizzes, graded by a mentor agent, Mr. Davis. In this open-ended system, students choose how to navigate a variety of learning sources, how to build their maps, and how often to quiz Betty. They may also interact with Mr. Davis, who can support their learning and teaching endeavors [10].

Betty's Brain is a suitable environment for examining SRL behaviors for two reasons. Firstly, students choose when and how to perform each step of the learning process (both their own and Betty's) [20, 33]. Indeed, the pedagogical agents in Betty's Brain are designed to facilitate the development of SRL behaviors by providing a framework for the gradual internalization of effective learning strategies. Secondly, students' interactions with Betty's Brain are logged to an online database with detailed timing information, enabling the microanalysis of student actions [37] for the measurement of SRL behaviors and strategies.

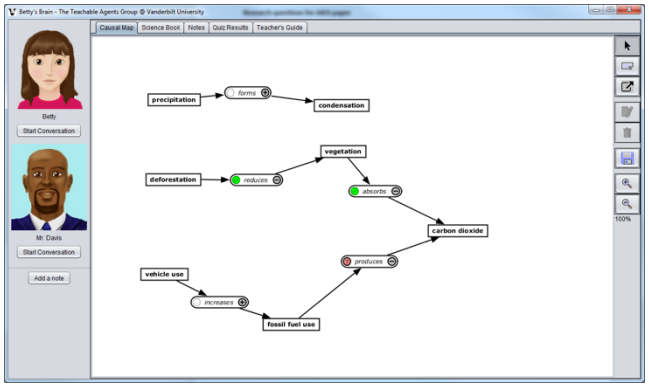


Figure 1. Screenshot of Betty's Brain showing a partial causal map constructed by a student.

2.2 Data Collection

This study examines data from 93 sixth graders who used Betty's Brain during their 2016–2017 science classes in an urban public school in Tennessee. The first data collection occurred over seven school days. On day 1, students completed a 30–45-minute paper-based pre-test that measured knowledge of scientific concepts and causal relationships. On day 2, students participated in a 30-minute training session about the learning goals and user interface. Afterwards (days 2–6), students used the Betty's Brain software for approximately 45–50 minutes each session, using concept maps to teach Betty about the causal relationships involved in the process of climate change. On day 7, students completed a post-test with the same questions as the pre-test. In addition to the data described, we also surveyed students on self-efficacy [31] and the task value [31].

A second data collection period occurred two months later, during which students were asked to model the causal relationships involved in thermoregulation. This was otherwise identical to the first session, but we consider only the learning data (pre – post test) from this second scenario (see section 4.2).

2.3 In-Situ Interviews

As students interacted with Betty's Brain, automatic detectors of educationally relevant affective states [19] and behaviors [26], already embedded in the software, identified key moments in the students' learning processes, either from specific affective patterns or theoretically aligned behavioral sequences. This detection was then used to prompt student interviews through Quick Red Fox (QRF), an app which integrates interview data with Betty's Brain events. Interviewers sought to take a helpful but non-authoritative role when speaking with students. Interviews were open-ended and occurred without a set script; however, students were often asked what their strategies were (if any) for getting through the system. As new information emerged in these open-ended interviews, questions were designed to elicit information about intrinsic interest (e.g., "What kinds of books do you like to read and why?"). Overall, however, students were encouraged to provide feedback about their experience with the software and talk about their choices as they used the software

2.4 Interview Transcription and Coding

A total of 358 interviews were conducted during this study and stored on a secure file management system. Interviews were manually transcribed by three members of the research team, preserving all metadata but scrubbing any identifying information.

The code development process followed [38]'s 7-stage recursive, iterative process: conceptualization, generation, refinement, codebook generation, revision and feedback, implementation, and continued revision. The conceptualization of codes involved a literature review to capture experiences relevant to affect and SRL. Using grounded theory [13], we worked with the lead interviewer (2nd author) to identify categories that were (1) theoretically valid and pertinent to the conditions in the COPES model and (2) likely to saliently emerge in the interviews.

We iteratively refined the coding scheme until the entire research team reached a shared understanding. Following the coding manual's production, external coders reached acceptable inter-rater reliability with the 3rd author before coding all of the transcripts. All codes had Cohen's kappa > .6, and the average Cohen's kappa across codes was .83. See Table 1 for details.

Table 1. Interview codes

Code	N	Description
Helpfulness	51	Utility of system resources for learning, and positive evaluations of the resources. κ=.643
Interestingness	11	Interestingness of system resources and continued desire to use the platform. κ=.726
Strategic Use	205	Indicates plan for interacting with the platform, or changes in strategy or interaction based on experiences. κ=.911
Positive Mr. Davis Attribution	8	Explicitly mentions interactions with Mr. Davis as positive experiences. κ=.838
Positive Science Attribution	26	Explicitly (positively) mentions science in relation to books, future careers, school subjects, and overall evaluations. κ=.837
Positive Persistence	105	Expression of a desire for challenge and that the current task is a challenge; there is active pursuit of a goal, and repeated attempts to complete a step/problem. κ=.911
Procedural Strategy	225	Step by step approach to the learning activity, active use of within-platform tools, reference to previous or upcoming step. κ=.862
Motivational Strategy	151	Explicit indication of expected outcome from behaviors/actions, explicitly mentions a pursuit for mastery, contains positive attribution/emotion for completion, and/or mentions desire to meet task demands. κ=.870
Self-Confidence	174	Positive description of own progress or ability, self-assessments of learning progress, willingness to encounter learning challenges/, recognition of helpful resources. κ=.877

2.5 SMART Encoding

We operationalized SRL behavior within the log data using the COPES and SMART SRL frameworks [42]. In this work, we categorize all student actions recorded in the log files as "operations" within the COPES model (defined as "cognitive and behavioral actions applied to perform the task"). We then evaluate these operations using the SMART model, which subcategorizes actions by the information taken as input and product generated [39]. Specifically, the SMART model presents five primitive cognitive operation subcategories: Searching, Monitoring, Assembling, Rehearsing, and Translating [39]. Each category is briefly described below; for more details, see [39, 41, 42, 45]. Examples specific to Betty's Brain are shown in Table 2.

Searching is the operation where a learner focuses their attention on a knowledge base or resource to update their working memory.

Monitoring considers two types of information: (1) learner perceptions (current understanding, quiz answers, etc.), (2) standards for performance. In monitoring activities, the learner evaluates their perceptions compared to the standards.

Assembling involves building a network of internal links between acquired information to understand relationships (X precedes Y, Y causes Z, etc.). Assembling activities help students to connect individual items of knowledge in working memory.

Rehearsing operations repeatedly direct attention to information that the learner is currently working on. These actions reinforce the same information and prevent decay in working memory.

Translating operations reformat information into a new representation, providing the potential for alternate interpretations and understanding. Examples include converting a diagram to plain text or answering a question about a diagram.

To enable a trace analysis of student SRL patterns [37] we first assigned each of the possible student operations within Betty's Brain to one SMART category. We categorized operations that added new items to the concept map within Betty's Brain as *assembling*, and operations that edited existing items as *monitoring*. In ambiguous cases, such as between translation and monitoring tasks, we considered student agency. Specifically, actions initiated by the system were classified as *translating* even if they had an evaluative component. In our operationalization of the SMART model, we found that Betty's Brain logged no rehearsing actions; thus, this category was not analyzed.

3. MODEL TRAINING METHODS

We built supervised machine learning models to detect each facet of the SMART model. We leveraged a combination of activity, survey, and interview data (described further below).

Table 2. Example Betty's Brain actions by SMART facet.

SMART Facet	N	Example
Searching	8	Searching the virtual textbook (initiated by the student)
Monitoring	22	Reviewing and updating the label of a causal link (initiated by the student)
Assembling	2	Adding a causal link to the map (initiated by the student)
Rehearsing	0	-
Translating	3	Responding to a system-initiated multiple-choice questions (vs. those initiated by the student)

3.1 Features

We split features into three groups based on their origin. Each group is described in detail below. Due to differences in scale, we *Z*-scored each feature prior to model training.

(Other) Student Activity Features (N = 4). These features provide a high-level description of student actions: the raw number of student actions, the proportion of links made that were ineffective, time spent off-task/idle (as characterized in [36]), and number of successful quizzes. These features were designed to be more coarse-grained than the log data used to derive the SMART variables. None of the fine-grained features used to calculate the SMART encoding are included in this feature set.

Student Interview Codes (N = 9). These were derived from the transcribed student interviews (described in section 2.3). In cases where students had multiple interviews, codes were averaged to provide one feature per code per student.

Survey Features (N = 2). Survey features come from the two survey measures described in section 2.2: self-efficacy and task value. While each measure consisted of multiple survey questions, both were summarized down to one variable, respectively.

3.2 Dependent Variables

We initially considered four dependent variables, the proportion of the time a student spent on each of the SMART variables discussed in section 2.5. We considered time spent rather than raw action counts for a more standardized comparison and to avoid misinterpretation. For example, there are more monitoring actions than searching actions; however, it is common for students to spend considerably more time searching than monitoring. Due to time spent idle (at least 30 seconds of inactivity [36]), the sum of these four variables for any given student may not be 1. The most common category was searching (M=0.65, SD=0.07), followed by monitoring (M=0.16, SD=0.06), translating (M=0.10, SD=0.02), and assembling (M=0.09, SD=0.04).

We also considered a second set of dependent variables related to student learning. We derived two variables, one for the current scenario from which the rest of the data was collected, and one for the future scenario. In both cases, learning was characterized by post test – pre test. We consider both scenarios to examine how well our approach generalizes to future interactions and understand how immediate context may influence prediction.

3.3 Regression Models

We used scikit-learn [28] to implement Bayesian ridge regression, linear regression, Huber regression, and random forest regression, and also implemented XGBoost with a separate library [15]. Hyperparameters were tuned on the training set using scikit-learn's cross-validated grid search [28] where appropriate.

All models were trained using 4-fold student-level cross-validation and repeated for ten iterations, each with a new random seed. For evaluation, predictions were pooled across folds, and averaged across iterations. These models then underwent a decision tree based secondary analysis, discussed below.

4. RESULTS

We compare model accuracy by computing the correlation between the model predictions and the ground truth values derived from student logs. We measured the Spearman *rho* correlation coefficient in the test folds to evaluate models. In the majority of cases, random forest regressors yielded the best results. As such, results from these models are reported below.

4.1 Predicting SMART Operations

We first consider results predicting the proportion of time a student spent on each of the four SMART operations. For each operation (i.e., searching, monitoring, assembling, and translating), we developed models drawn from various combinations of our feature types (actions, surveys, and interview codes). Thus, we were able to test the modeling potential of seven different combinations of features for each SMART operation (see Table 3). To provide a point of comparison, we generated a chance baseline for each variable by shuffling the ground truth values. This allowed us to estimate a random baseline that still preserved the original distribution.

Table 3. Spearman correlations predicting ground truth labels of self-regulated learning operations

Features	Searching	Monitoring	Assembling	Translating
Chance Baseline	0.01	0	0.01	0
Individual Feature Sets				
Student Surveys (Surveys)	0.28	0.29	0.28	0.08
Student Interviews (Int)	0.31	0.37	0.35	0.09
Student Actions (Act)	0.27	0.47	0.59	0.11
Combined Feature Sets				
Int + Surveys	0.35	0.42	0.62	0.13
Act + Surveys	0.29	0.47	0.63	0.12
Act + Int	0.34	0.51	0.64	0.1
Act + Int + Surveys	0.39	0.55	0.66	0.19

We note that all models outperformed baseline, and that models consistently performed worst at predicting Translating. This may be due to the low variance between students as noted above. We note that the best model performance was achieved by combining the three feature sets (Actions + Interviews + Surveys). This suggests that even though these operations are derived from student log data, additional context from interviews and surveys can improve SRL predictions.

4.1.1 Feature Interaction Analysis

Our most successful models were tree-based, meaning that they may contain nonlinear relationships that would be unsuitable for linear feature analysis. Therefore, we trained one decision tree regressor per outcome and examined each tree's top two levels to observe the most important interactions, each of which was classified as "High" or "Low."

As Table 4 shows, Self-Confidence and Self-Efficacy frequently occur in these interactions, implying students' self-regulation

Table 4. Top 8 interactions for predicting SMART facets

Feature 1			Feature 2			Predicted Value	
Low	Self-Confidence	+	Low	Successful Quizzes	=	High	Searching
High	Self-Confidence	+	Low	Off Task Time	=	Low	Searching
Low	Off Task Time	+	Low	Self-Efficacy	=	High	Monitoring
High	Off Task Time	+	High	Self-Efficacy	=	High	Monitoring
Low	Action Count	+	Low	Self-Efficacy	=	High	Assembling
High	Action Count	+	Low	Ineffective Links	=	Low	Assembling
Low	Procedural Strategy	+	Low	Self Confidence	=	Low	Translating
High	Procedural Strategy	+	Low	Motivational Strategy	=	High	Translating

hinged on their perception of themselves. For example, students with high Self-Confidence who spent less time off task were still likely to have lower searching values, ostensibly because they may not feel the need to consult external resources.

4.2 Predicting Student Learning

Next, we explored how the four SMART facets predicted student learning (operationalized as post-test – pre-test) in both the *current* scenario (from which all the data used in the models was collected) and then the *future* scenario (collected in a second round of data collection with the same students; see section 2). In this future scenario, the content was different (climate change vs. thermoregulation), but the software remained the same.

We consider three feature sets: 1) the three feature sets used in section 4.1 combined; 2) the ground truth values for the SMART encodings (dependent variables in section 4.1); 3) predicted values for each of the SMART operations generated using the best models from section 4.1.

For both learning outcomes, we tested both the Ground Truth values collected from the first scenario (i.e., the actual searching or monitoring behaviors from that scenario) and Predicted SMART values (as predicted by the Act + Int + Survey models from the current scenario). This allowed us to examine how data collected in the current scenario generalizes to a future learning session.

As Table 5 shows, each learning model outperformed chance, demonstrating both predictive validity and generalizability. These results also present two findings of note. Firstly, learning models constructed from Predicted SMART values outperformed those constructed from the Ground Truth SMART values for both scenarios. It is possible that our models in fact, smooth over some of the noise that is present in the ground truth, thus presenting a more robust measure than the raw encodings [6].

Second, we note that for the future scenario, the predicted SMART values outperform model constructed directly from the Act + Int + Survey variables, despite this being the values from which the SMART predictions are made. The SMART values may provide a latent encoding of this data, which is more generalizable than the raw values to future occurrences, however further study would be required to confirm this hypothesis.

Table 5. Spearman rho for models predicting learning gains. All features are derived from the current scenario

Features	Current Scenario	Future Scenario
Chance Baseline	.01	.01
Act + Int + Survey	.45	.37
Ground Truth SMART	.21	.29
Predicted SMART	.32	.43

4.2.1 Feature Interaction

Using the same feature analysis methods described in section 4.1.1 we again examined the interactions involved when predicting learning gains. These results are shown in Table 6.

We note the need for the balance between SMART operations. For example, *high* monitoring and low translating resulted in lower learning on the current scenario, but so did high searching with *low* monitoring, suggesting it would be insufficient to simply increase monitoring activities; we must encourage more effective combinations of operations. Similarly, these results imply the need for a careful structure approach to assembling.

The results shown for the future scenario focus on more transferrable features than results for the current scenario. This makes sense given that we are no longer considering the immediate context. We found that students who had low off task time and high persistence in the first scenario were more likely to perform well in the second. Students with lower monitoring but high translating were likely to have lower learning, indicating it is not enough to simply test your knowledge, it is also important to review feedback and compare work to standards.

5. DISCUSSION

Adaptive learning technology that responds to students' learning patterns can improve both immediate and long-term goals by supporting the internalization of appropriate self-regulated learning behaviors. In this paper, we infer SRL using a combination of data mining and interviews/surveys.

5.1 Main Findings

Automated detection of SRL behaviors poses several challenges, as many of the processes it entails are highly internal [42]. In this work, we demonstrate that a combination of activity data, data from surveys, and student interviews provides a more robust prediction of SRL than any individual data stream. We find that predicted SRL behaviors (from students' first system interactions) predict future performance. In fact, models based on our inferred SRL measures outperform models constructed from the original features used to train them (action, interview, and survey data) and the SMART ground truth values. This finding is important for environments where detailed trace analysis may not be possible, but coarser-grained activity can be distilled.

Further, we show that a balanced combination of SRL behaviors is required for successful learning. For example, students with low learning are likely not spending enough time monitoring, but simply requiring them to check their work more often may not create improvement if they have not yet fully assembled the knowledge necessary to effectively examine their previous efforts. Future work should design scaffolds to create this balance.

Table 6. Top interactions for predicting learning. * indicates a predicted value

Scenario	Feature 1			Feature 2			Predicted Value	
Current Scenario	High	Monitoring*	+	Low	Translating*	=	Low	Learning
	High	Searching	+	Low	Monitoring	=	Low	Learning
	High	Successful Quizzes	+	High	Self-Efficacy	=	High	Learning
	Low	Successful Quizzes	+	High	Ineffective Links	=	Low	Learning
Future Scenario	High	Monitoring*	+	Low	Assembling*	=	High	Learning
	Low	Monitoring*	+	Low	Assembling*	=	Low	Learning
	Low	Off Task Time	+	High	Positive Persistence	=	High	Learning
	Low	Monitoring	+	High	Translating	=	Low	Learning

These results demonstrate the importance of considering log data in the context of other measures when understanding student SRL. This, in turn, underscores the need for more automated measures of complex noncognitive measures such as self-efficacy and persistence. Our work shows that these codes collected from interview data boost SRL detection. In order to scale SRL detection, we must first consider how we might automate the detection of some of the constructs discussed here (see future work below). These results offer the potential for designing pre-emptive interventions, providing a more informed, asset-based intervention as opposed to responding to a negative event.

5.2 Applications

The key application of this work is to develop adaptive online learning environments that respond to student SRL. As SRL detection continues to improve, systems like Betty's Brain might choose from wide range of intervention strategies that have already been shown to improve SRL (e.g., discussion in section 1). For example, once students who are not employing optimal strategies have been identified, additional scaffolding tasks might be used to encourage new behaviors. Similarly, the software could deliver interventions to increase motivation or interest.

It is important to note that the proposed intervention strategies rely on SRL detection, which is likely always to be imperfect. Self-regulation is highly internal [32], and as such, it is unlikely that we will ever be able to infer SRL perfectly. Any interventions should be designed to be "fail-soft" in that there are no damaging effects to student learning or future SRL if delivered incorrectly.

In situations where computer-based learning is being used to augment classroom instruction, a further application of this work would be in providing feedback to teachers. Such feedback could help them dynamically adapt their instruction, as outlined in [18] for example, providing real-time feedback or an early warning system, etc.

5.3 Limitations and Future Work

This work has limitations that should be addressed going forwards. Firstly, the SMART features only characterize student operations, and they do not give a complete SRL picture. Future work should look to combine the SMART framework with the broader COPES model [43]. The interview and survey measures used in this work may also capture aspects of the cognitive and task conditions referred to in the COPES model, but additional study would be required to confirm this hypothesis.

A further limitation is the slightly cyclic nature of using student activity features derived from log data, to predict SRL, also derived from log data. While we made every effort to ensure that our models were not confounded in some way, future work should consider an external measure of SRL for additional validation [44].

Finally, interview data is time-consuming to collect, limiting scalability. In the future, we will employ alternate measures for some of the interview codes measured in this work, such as student surveys. It is possible that voice recognition and natural language processing could be used in the future to support this type of data collection.

5.4 Conclusions

This paper investigates predicting student SRL behavior in a computer-based learning environment from a complex dataset of coarse-grained activity data, in-situ student interviews, and student surveys. Our analyses indicated that SRL was best predicted from a combination of the three feature sets. We found our predicted SRL operations were better at predicting future learning than their ground truth equivalents, suggesting the potential for a smoother latent encoding and better supporting students in future endeavors. We envision this paper contributing to future technologies that will track and respond to student SRL behaviors and create more positive learning experiences.

6. ACKNOWLEDGMENTS

This work was supported by NSF #DRL-1561567.

7. REFERENCES

[1] Azevedo, R., Johnson, A., Chauncey, A. and Burkett, C. 2010. Self-Regulated Learning with MetaTutor: Advancing the Science of Learning with MetaCognitive Tools BT - New Science of Learning: Cognition, Computers and Collaboration in Education. (2010), 225–247.https://doi.org/10.1007/978-1-4419-5716-0_11.

[2] Azevedo, R., Johnson, A., Chauncey, A. and Graesser, A. 2015. Use of Hypermedia to Assess and Convey Self-Regulated Learning. *Handbook of Self-Regulation of Learning and Performance*. 10622 (2015).https://doi.org/10.4324/9780203839010.ch7.

[3] Azevedo, R., Moos, D.C., Johnson, A.M. and Chauncey, A.D. 2010. Measuring Cognitive and Metacognitive Regulatory Processes during Hypermedia Learning: Issues and Challenges. *Educational Psychologist*. (2010).https://doi.org/10.1080/00461520.2010.515934.

[4] Azevedo, R. and Strain, A.C. 2011. Integrating Cognitive, Metacognitive, and Affective Regulatory Processes with MetaTutor. *New Perspectives on Affect and Learning Technologies*. (2011).https://doi.org/10.1007/978-1-4419-9625-1_11.

[5] Azevedo, R., Witherspoon, A., Chauncey, A., Burkett, C.

and Fike, A. 2009. MetaTutor: A MetaCognitive Tool for Enhancing Self-Regulated Learning. *AAAI Fall Symposium - Technical Report*. FS-09-02, (2009), 14–19.

[6] Baker, R.S.J. d., Corbett, A.T. and Aleven, V. 2008. Improving Contextual Models of Guessing and Slipping with a Truncated Training Set. (2008).

[7] Bannert, M. and Reimann, P. 2012. Supporting Self-Regulated Hypermedia Learning through Prompts. *Instructional Science*. 40, 1 (2012), 193–211.

[8] Biswas, G., Baker, R.S. and Paquette, L. 2017. Data Mining Methods for Assessing Self-Regulated Learning. *Handbook of Self-Regulation of Learning and Performance*. (2017), 388–403.https://doi.org/10.4324/9781315697048-25.

[9] Biswas, G., Leelawong, K., Schwartz, D., Vye, N., Davis, J., Belynne, K., Viswanath, K., Brandsford, J. and Katzlberger, T. 2005. Learning by Teaching: A New Agent Paradigm for Educational Software. *Applied Artificial Intelligence*. (2005).https://doi.org/10.1080/08839510590910200.

[10] Biswas, G., Segedy, J.R. and Bunchongchit, K. 2016. From Design to Implementation to Practice a Learning by Teaching System: Betty's Brain. *International Journal of Artificial Intelligence in Education*. (2016).https://doi.org/10.1007/s40593-015-0057-9.

[11] Boekaerts, M. and Pekrun, R. 2016. Emotions and Emotion Regulation in Academic Settings. *Handbook of Educational Psychology*. (2016), 76–90.https://doi.org/10.1017/S0954579400006301.

[12] Broadbent, J. and Poon, W.L. 2015. Self-Regulated Learning Strategies & Academic Achievement in Online Higher Education Learning Environments: A Systematic Review. *Internet and Higher Education*. 27, (2015), 1–13.https://doi.org/10.1016/j.iheduc.2015.04.007.

[13] Charmaz, K. 1983. The Grounded Theory Method: An Explication and Interpretation. *Contemporary Field Research*. (1983), 109–126.

[14] Chen, C.-M. and Huang, S.-H. 2014. Web-Based Reading Annotation System with an Attention-Based Self-Regulated Learning Mechanism for Promoting Reading Performance. *British Journal of Educational Technology*. 45, 5 (2014), 959–980.

[15] Chen, T. and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2016), 785–794.https://doi.org/10.1145/2939672.2939785.

[16] Cleary, T.J. and Zimmerman, B.J. 2012. A Cyclical Self-Regulatory Account of Student Engagement: Theoretical Foundations and Applications. *Handbook of Research on Student Engagement*. (2012).https://doi.org/10.1007/978-1-4614-2018-7_11.

[17] Dignath, C., Buettner, G. and Langfeldt, H.-P. 2008. How Can Primary School Students Learn Self-Regulated Learning Strategies Most Effectively?: A Meta-Analysis on Self-Regulation Training Programmes. *Educational Research Review*. 3, 2 (2008), 101–129.https://doi.org/https://doi.org/10.1016/j.edurev.2008.02.003.

[18] Holstein, K., McLaren, B.M. and Aleven, V. 2017. Intelligent tutors as teachers' aides: exploring teacher needs for real-time analytics in blended classrooms. *Proceedings of the Seventh International Learning Analytics & Knowledge Conference* (2017), 257–266.

[19] Jiang, Y., Bosch, N., Baker, R.S., Paquette, L., Ocumpaugh, J., Andres, J.M.A.L., Moore, A.L. and Biswas, G. 2018. Expert feature-engineering vs. Deep neural networks: Which is better for sensor-free affect detection? *Artificial Intelligence in Education* (2018), 198–211.https://doi.org/10.1007/978-3-319-93843-1_15.

[20] Kinnebrew, J.S., Biswas, G., Sulcer, B. and Taylor, R.S. 2013. Investigating Self-Regulated Learning in Teachable Agent Environments. *International Handbook of Metacognition and Learning Technologies.* (2013), 451–470.https://doi.org/10.1007/978-1-4419-5546-3_29.

[21] Kinnebrew, J.S., Segedy, J.R. and Biswas, G. 2014. Analyzing the Temporal Evolution of Students' Behaviors in Open-Ended Learning Environments. *Metacognition and Learning*. (2014).https://doi.org/10.1007/s11409-014-9112-4.

[22] Kizilcec, R.F., Pérez-Sanagustin, M. and Maldonado, J.J. 2017. Self-Regulated Learning Strategies Predict Learner Behavior and Goal Attainment in Massive Open Online Courses. *Computers & Education*. 104, (2017), 18–33.

[23] Klug, J., Ogrin, S. and Keller, S. 2011. A Plea for Self-Regulated Learning as a Process : Modelling , Measuring and Intervening. *Psychological Test and Assessment Modeling*. 53, 1 (2011), 51.

[24] de la Fuente, J., Sander, P., Kauffman, D.F. and Yilmaz Soylu, M. 2020. Differential Effects of Self- vs. External-Regulation on Learning Approaches, Academic Achievement, and Satisfaction in Undergraduate Students. *Frontiers in Psychology*. 11, (2020), 2678.https://doi.org/10.3389/fpsyg.2020.543884.

[25] Lee, H.W., Lim, K.Y. and Grabowski, B.L. 2010. Improving Self-Regulation, Learning Strategy Use, and Achievement with Metacognitive Feedback. *Educational Technology Research and Development*. 58, 6 (2010), 629–648.

[26] Munshi, A., Rajendran, R., Ocumpaugh, J., Biswas, G., Baker, R.S. and Paquette, L. 2018. Modeling learners' cognitive and affective states to scaffold srl in open-ended learning environments. *UMAP 2018 - Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization* (2018), 131–138.https://doi.org/10.1145/3209219.3209241.

[27] Pandey, A., Hale, D., Das, S., Goddings, A.-L., Blakemore, S.-J. and Viner, R.M. 2018. Effectiveness of Universal Self-Regulation–Based Interventions in Children and Adolescents: A Systematic Review and Meta-Analysis. *JAMA Pediatrics*. 172, 6 (Jun. 2018), 566–575.https://doi.org/10.1001/jamapediatrics.2018.0232.

[28] Pedregosa, F. et al. 2011. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*. 12, (2011), 2825–2830.https://doi.org/10.1007/s13398-014-0173-7.2.

[29] Pérez-Álvarez, R., Maldonado-Mahauad, J. and Pérez-Sanagust\'\in, M. 2018. Tools to support self-regulated learning in online environments: literature review. *European conference on technology enhanced learning* (2018), 16–30.

[30] Pintrich, P.R. 2000. The Role of Goal Orientation in Self-Regulated Learning. *Handbook of Self-Regulation*. (2000), 451–502.https://doi.org/10.1016/b978-012109890-2/50043-

3.

[31] Pintrich, P.R.R., Smith, D., Garcia, T. and McKeachie, W. 1991. A Manual for the Use of the Motivated Strategies for Learning Questionnaire (MSLQ). *Ann Arbor*. *Michigan*. (1991).https://doi.org/ED338122.

[32] Puustinen, M. and Pulkkinen, L. 2001. Models of Self-regulated Learning: A review. *Scandinavian Journal of Educational Research*.

[33] Roscoe, R.D., Segedy, J.R., Sulcer, B., Jeong, H. and Biswas, G. 2013. Shallow Strategy Development in a Teachable Agent Environment Designed to Support Self-Regulated Learning. *Computers and Education*. (2013).https://doi.org/10.1016/j.compedu.2012.11.008.

[34] Schraw, G. 2010. Measuring Self-Regulation in Computer-Based Learning Environments. *Educational Psychologist*. (2010).https://doi.org/10.1080/00461520.2010.515936.

[35] Schwartz, D.L., Chase, C., Chin, D.B., Oppezzo, M., Kwong, H., Okita, S., Biswas, G., Roscoe, R., Jeong, H. and Wagster, J.D. 2009. Interactive Metacognition: Monitoring and Regulating a Teachable Agent. *Handbook of Metacognition in Education*. (2009), 340–358.

[36] Segedy, J.R., Kinnebrew, J.S. and Biswas, G. 2015. Using Coherence Analysis to Characterize Self-Regulated Learning Behaviours in Open-Ended Learning Environments. *Journal of Learning Analytics*. (2015).https://doi.org/10.18608/jla.2015.21.3.

[37] Siadaty, M., Gasevic, D. and Hatala, M. 2016. Trace-Based Microanalytic Measurement of Self-Regulated Learning Processes. *Journal of Learning Analytics*. 3, 1 (2016), 183–214.https://doi.org/https://doi.org/10.18608/jla.2016.31.11.

[38] Weston, C., Gandell, T., Beauchamp, J., McAlpine, L., Wiseman, C. and Beauchamp, C. 2001. Analyzing Interview Data: The Development and Evolution of a Coding System. *Qualitative Sociology*. 24, 3 (2001), 381–400.https://doi.org/10.1023/A:1010690908200.

[39] Winne, P.H. 2011. A Cognitive and Metacognitive Analysis of Self-Regulated Learning. *Handbook of Self-Regulation of Learning and Performance*. (2011), 15–32.https://doi.org/10.4324/9780203839010.ch2.

[40] Winne, P.H. 2010. Improving Measurements of Self-Regulated Learning. *Educational Psychologist*. 45, 4 (2010), 267–276.https://doi.org/10.1080/00461520.2010.517150.

[41] Winne, P.H. 2014. Issues in Researching Self-Regulated Learning as Patterns of Events. *Metacognition and Learning*. 9, 2 (2014), 229–237.https://doi.org/10.1007/s11409-014-9113-3.

[42] Winne, P.H. 2017. Learning Analytics for Self-Regulated Learning. *Handbook of Learning Analytics*. (2017), 241–249.https://doi.org/10.18608/hla17.021.

[43] Winne, P.H. and Hadwin, A.F. 1998. Studying as Self-Regulated Learning. *Metacognition in Educational Theory and Practice*. (1998), 277–304.

[44] Winne, P.H. and Perry, N.E. 2000. Measuring Self-Regulated Learning. *Handbook of Self-Regulation*. (2000).https://doi.org/10.1016/b978-012109890-2/50045-7.

[45] Winne, P.H., Teng, K., Chang, D., Lin, M.P.C., Marzouk, Z., Nesbit, J.C., Patzak, A., Rakovic, M., Samadi, D. and Vytasek, J. 2019. NStudy: Software for Learning Analytics about Learning Processes and Self-Regulated Learning. *Journal of Learning Analytics*. (2019).https://doi.org/10.18608/jla.2019.62.7.

[46] Wong, J., Baars, M., Davis, D., Van Der Zee, T., Houben, G.J. and Paas, F. 2019. Supporting Self-Regulated Learning in Online Learning Environments and MOOCs: A Systematic Review. *International Journal of Human-Computer Interaction*. 35, 4–5 (2019), 356–373.https://doi.org/10.1080/10447318.2018.1543084.

[47] Zimmerman, B.J. 1989. Models of Self-Regulated Learning and Academic Achievement. *Self-Regulated Learning and Academic Achievement*. (1989), 1–25.https://doi.org/10.1007/978-1-4612-3618-4_1.

The effects of a personalized recommendation system on students' high-stakes achievement scores: A field experiment

Nilanjana Chakraborty
Department of Statistics
University of Florida
nchakraborty@ufl.edu

Samrat Roy
Department of Statistics
University of Florida
samratroy@ufl.edu

Walter L. Leite
College of Education
University of Florida
walter.leite@coe.ufl.edu

Mohamad Kazem Shirani Faradonbeh
Department of Statistics
University of Georgia
mohamadksf@uga.edu

George Michailidis
Department of Statistics and the Informatics Institute
University of Florida
gmichail@.ufl.edu

ABSTRACT

This study examines data from a field experiment investigating the effects of a personalized recommendation algorithm that proposes to students which videosto watch next, after they complete mini-assessments for algebra that available on the Math Nation intelligent virtual learning environment (IVLE). The end users of Math Nation are students enrolled in an Algebra 1 course in middle and high schools of the state of Florida, and the IVLE is used both during and out of school time. The objective of the developed recommendation algorithm is to increase student preparation to take the state-mandated End-of-Course (EoC) Algebra 1 assessment at the end of the school year. The algorithm is based on a Markov Decision Process framework that uses as input the students' responses to a series of mini-assessment tests. The current study randomly assigned 16,406 students to either treatment or control conditions, which were blind to both students and teachers. The results indicate that the effects of the recommendation algorithm depend on the level of usage of students, showing significant improvements on EoC test scores of students who have a moderate level of usage. However, there was no effect for low usage students. The study also shows that students practicing with the mini-assessments available on Math Nation, helps them improve by a small margin their performance on the End-of-Course test, irrespective of the usage level. Finally, the study provides insights on challenges posed for implementing personalized recommendation algorithms at a large scale, related both to student self-regulation and teacher orchestration of technology use in the classroom.

Nilanjana Chakraborty, Samrat Roy, Walter Leite and George Michailidis "The effects of a personalized recommendation system on students' high-stakes achievement scores: A field experiment". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 588-594. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education

Keywords

Personalized Recommendation, Randomized Control Study, Hierarchical Clustering, Markov Decision Process, Algebra

1. INTRODUCTION

There is a growing trend in employing intelligent virtual learning environments (IVLE) to aid students in improving their math performance in K-12 education [8, 26, 23]. While there is a robust body of literature that shows that students' preparedness together with various demographic and school characteristics are key factors for predicting students' performance in various math tests [15], IVLE have been viewed as an especially promising way of improving students' achievements in mathematics. Given the investment of resources into technology products and the time and effort needed to integrate them into the curriculum, there has been considerable interest in determining their effectiveness. A number of studies have reported positive effects based both on small scale randomized control trials and longer term interventions [14, 13, 19, 16, 15], as well as based on observational data [12]. There have also been a series of meta-analysis studies showing that IVLE have substantial effects on student outcomes [10, 11, 24, 27].

IVLE have the potential of offering personalized learning experiences. The latter refer to instruction "in which the pace of learning the instructional approach are optimized for the needs of each learner", according to the United States National Education Technology Plan 2017. IVLE that offer some degree of personalization include Khan Academy at the K-12 level and Newton at the higher education level. As discussed in [3], at the core of personalized learning strategies is a recommendation algorithm aiming to propose appropriate learning materials and topics to the student at the right time, leveraging the student's prior history of interactions with the IVLE.

Many personalized learning strategies leverage ideas and tools from the field of Reinforcement Learning [4, 5, 9, 20]. The key components of a reinforcement learning based algorithm are the triplet of *state, reward, action*. The state reflects information on the student's knowledge and skills set on the topic(s) under consideration, the reward relates to the goals of the strategy (e.g. performance on tests, engagement with the IVLE, etc.) and the action refers to an activity, (e.g. watch a video on a topic of interest, take an assessment test, etc.) that, based on the current state information, aims to maximize the expected reward.

This study reports the results of a large-scale randomized field experiment that focuses on the impact of a simple personalized strategy implemented on the Math Nation IVLE, on a high-stakes, state-mandated End-of-Course (EoC) algebra test. Although many evaluations of IVLE have been published, most of them rely on locally developed standardized tests, rather than high-stakes statewide tests [10]. Math Nation, is an online video-based tutoring program aiming to prepare students in the state of Florida for the EoC, which is required for high school graduation. The platform offers videos on various algebra topics recorded by different tutors, explaining the main concepts and walking the student through related examples, 3-question assessments for each topic and 10-question assessments for sets of related topics, with video explanations for each question. Therefore, students can assess their progress by taking both the short (3-question) and the long (10-question) tests. Further, the platform offers a monitored discussion area, wherein students can pose questions to peers and volunteer tutors. Hence, at launch time, it shared a number of characteristics with Khan Academy, both being self-guided and easy to use on an ad hoc basis, without the need for extensive professional development training for teachers. The content of the videos and assessments are aligned with the curriculum adopted by the state and also the content and format of the EoC test.

A new feature of Math Nation is the introduction of an algorithm to recommend videos to students, leveraging information on their performance on the mini-assessments associated with each video. Specifically, Math Nation divides the whole Algebra 1 course materials into 10 sections. Each section is further divided into several topics, thus resulting in a total of 93 topics for the entire course. For each topic, there is a tutorial video associated with it, recorded by different tutors. At the end of the video the student is presented with a 3-question assessment (henceforth called a mini-assessment) and based on the score obtained, a video recommendation (the action) is offered aiming to maximize the student's expected score (the reward) on these mini-assessments. The student can follow the recommendation or decide to ignore it and select another video of her/his own choice by the same or another tutor. To compare the effectiveness of the recommendation algorithm, a "business-as-usual" competitor is implemented, which recommends the next video in a predetermined sequence related to the structure of the algebra state curriculum, irrespective of the score achieved in the mini-assessment.

The objectives of the study are twofold: (i) estimate the average treatment effect of the recommendation algorithm vis-a-vis its competitor together with its interactions with previous achievement and level of usage of the algorithm, and (ii) understand the relationship between performance in the mini-assessments and the EoC test, after accounting for math preparedness and school characteristics of the students that participated in this randomized control study.

The remainder of the paper is structured as follows. Section 2 presents the developed personalized recommendation strategy. Section 3 describes in detail the data recorded from the algorithm, as well as other covariates used in the analysis. Section 4 presents the statistical methods used in the analysis and the main results of the study. Finally, Section 5 discusses the implications of our findings and suggestions to modify the recommendation algorithm.

2. PERSONALIZED RECOMMENDATION STRATEGY

Next, we describe the data-driven algorithm for recommending a suitable tutoring video to each individual student. As previously mentioned, the content of the course is divided into 93 topics, with each topic accompanied by a video recorded by 5 tutors in English and 1 tutor in Spanish. Students can freely select the tutor for each video.

To rigorously set the stage for the video recommendation algorithm, fix a single student, and let $s_k(t)$ be the corresponding "mini-score" for topic $k \in \{1, 2, \cdots, 93\}$, at time $t = 0, 1, \cdots$. These mini-scores, representing the knowledge level of the student, are obtained by assessing responses to the mini-assessments comprising of 4-choice questions, with a single correct choice. Thus, the set of possible outcomes consists of i correct answer(s), together with $3-i$ wrong answer(s), for $i = 0, 1, 2, 3$. Then, we center and normalize the corresponding scores (henceforth referred to as mini-scores), so that on average, simply guessing the answers lead to a zero score. Thus, we have $s_k(t) \in \{-3, 1, 5, 9\}$, and if the answers are selected completely at random, $\mathbb{E}[s_k(t)] = 0$.

With the above setting, the full state of the student at time t is given by $S(t) = [s_1(t), \cdots, s_{93}(t)]' \in \{-3, 1, 5, 9\}^{93}$, while $\|S(t)\| = \sum_{k=1}^{93} s_k(t)$ reflects the (total) score of the student under consideration at time t. The dynamical model for topic k consists of a Markov chain for which the state is $s_k(t)$. For the time being, suppose that the parameters of the Markov chain consisting of 4×4 tables of transition probabilities among the states $\{-3, 1, 5, 9\}$ are available. We will shortly discuss a statistical method leveraging transfer learning techniques, for estimating the Markov transition kernels according to the observed data.

The recommendation strategy is to propose to the student the tutoring video corresponding to the topic with the *largest predicted growth* in the mini-score. Formally, at time t, the IVLE recommends the student to watch the tutoring video of topic $k^\star$, wherein

$$k^\star = \arg\max_k \mathbb{E}\left[s_k(t+1) - s_k(t) \,\middle|\, S(t)\right],$$

where the notation " | " is used to indicate a conditional probability distribution. The student can either accept the recommendation, watch the video and take the mini-assessment, or can ignore the recommendation and select another video

to watch (by possibly another tutor).

Note that in order to compute the above expected values, for every topic $i \in \{1, \cdots, 93\}$ it suffices to have only the 4 probabilities corresponding to the transition of the Markov chain from the current state $s_i(t)$ to the next one $s_i(t+1)$. Intuitively, the difference quantity $s_k(t+1) - s_k(t)$ reflects the predicted growth of the student in topic k. Therefore, the high level idea of the recommendation strategy is to propose to the student to work on the topic (s)he is capable of improving her/his knowledge level the most. Therefore, the recommendation aligns with Vygotsky's theory [28] of zone of proximal development by providing a video that is neither too easy, nor too challenging. Further, the recommended topic is totally personalized to the student, since the state $S(t)$ at time t is unique to each student.

Finally, we describe the statistical learning procedure for estimating the Markov transition probabilities. For this purpose, the students are clustered in 12 different groups, based on their demographic and other background data, so that students of similar learning abilities will be assigned to the same group (cluster). The details of the clustering procedure are provided in Section 3. We assume that students in each group share the Markov transition probabilities reflecting their cognitive responses to watching the tutoring video of a specific topic. Thus, in order to estimate the transition probabilities for students in a fixed group, we divide the total number of transitions between every pair of the possible states $\{-3, 1, 5, 9\}$ in the group, with the total number of transitions in the group. We emphasize the following points. First, while the Markov transition probabilities are the same for all students in one demographic/background group, the states are uniquely personalized to each student. Second, the estimates of the transition probabilities change over time as the platform collects more data from the responses of the students to the mini-assessments. Further, when Math Nation starts being used by the students, the initial estimates of the transition probabilities are selected randomly, and are updated throughout the academic year as the students continue to use it. Finally, if there is more than one $k^\star$ maximizing the predicted growth, one will be selected at random.

Before the algorithm was deployed within Math Nation platform, it was extensively tested on synthetic data generated based on data collected in previous years from the platform. Specifically, students that have used the platform in previous years were clustered in 12 groups (see also Section 3) based on their demographic and background information. Note that the distributions of such data are very similar to those in the academic year that the recommendation algorithm was launched and evaluated in the current study. Subsequently, the response data to the mini-assessment tests of the students within each cluster were used to estimate the corresponding Markov transition probabilities. The latter were then used to initialize the recommendation algorithm and to generate synthetic data for students in different clusters. The upshot of this analysis was that the algorithm required adequate engagement ($t \geq 45$) to show significant improvement in performance in the mini-assessments. We revisit this point in the Discussion section.

Table 1: Distribution of the students across different Math Achievement Levels, School Grades and Student Grades

Achievement Level	No. of Students	School Grades	No. of Students	Student Grades	No. of Students
1	473	A	4,377	5	3
2	1,453	B	2,001	6	1463
3	3,487	C	4580	7	3599
4	2,711			8	5893
5	2,834				
Total	10,958	Total	10,958	Total	10,958

Note: Data based on previous school year performance

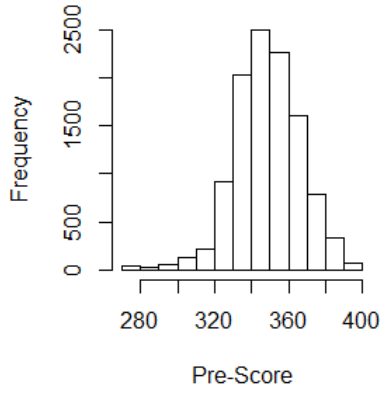


Figure 1: Distribution of Pre-Score

3. DATA DESCRIPTION

In this study, we randomly assign a sample of 16,406 middle and high school students enrolled in Algebra 1 in a large school district in the state of Florida, to a treatment (proposed recommendation strategy) or a control (business as usual recommendation strategy) group. The assignment was blind to students and teachers. The treatment group received video recommendations as described in the previous section, while the control group received a recommendation to watch the next video in the curriculum sequence. To initialize the recommendation, a randomized cluster design was employed. Specifically, students were first matched according to their grade, school characteristics and math preparedness test scores from the previous school year and then randomly assigned to the two groups. The variables used for matching purposes were the scores on the state standardized mathematics test, called the Mathematics Florida Standards Assessment[1](henceforth, referred to as Pre-Score and the corresponding test referred to as Pre-Test), as well as an achievement level assigned to them by their schools, while the quality of each school is reflected by a grade assigned to it by the state Department of Education[2]. The latter grades are based on several components and have five different levels ('A' being the highest level and 'F' being the lowest one). Due to lack of data for many of these variables, 5,448 students were removed from any further analysis and hence Table 1 that shows the distributions of the students across different Achievement Levels, School Grades and Student Grades and Figure 1 that depicts the distribution of the Pre-Score are based on the remaining 10,958 students.

[1]http://www.fldoe.org/accountability/assessments/k-12-student-assessment/fsa.stml

[2]http://www.fldoe.org/accountability/accountability-reporting/school-grades/

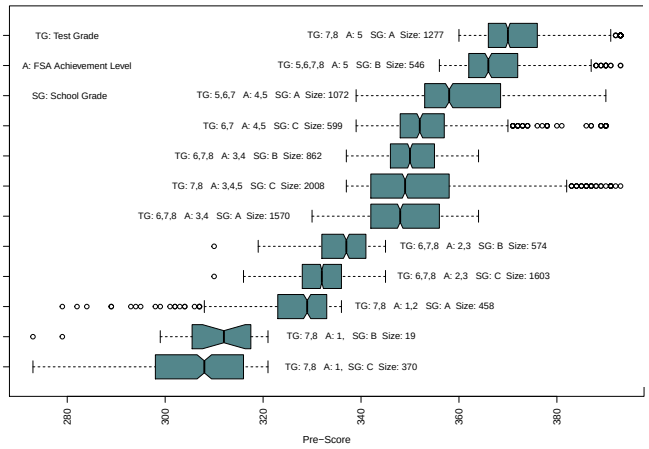


Figure 2: Boxplots of clusters hierarchically ordered based on Pre-score: For each cluster, School Grade, Achievement Level and Student Grade and Cluster Size are reported.

Using the above information, students were assigned to clusters/groups. This cluster assignment is used as a categorical variable in the analysis presented in Section 4. The clusters are designed in such a way that each of them corresponds to a group with a unique combination of math preparedness and school grade. In summary, the following four variables were considered by the clustering algorithm: Pre-Score, Math Achievement Level, School Grade and Student Grade. An agglomerative hierarchical clustering algorithm was employed for this task and using the dendrogram with Gower's distance metric, along with silhouette values [7], the number of clusters was chosen to be 12. Figure 2 provides a pictorial representation of the key features of the clusters. Specifically, for each cluster the Figure depicts the boxplot of the Pre-Score and also the corresponding Student Grade, Math Achievement Level, School Grade and Size of the cluster. For ease of comparison, the clusters are ordered according to the distribution of the Pre-Score. Hence, cluster 1 corresponds to the group of students having the lowest Pre-Score, while cluster 12 is the group with the highest Pre-Score. As Figure 2 shows, the size of cluster 2 was very small and hence it was merged with cluster 1 for the subsequent analyses.

The number of times a particular student takes the mini-assessment after watching a video, is defined as the usage by that student. Figure 3 depicts the average usage per student for each of the clusters for both the control and treatment groups. It can readily be seen that the overall average across the study population is 2.88, with many clusters exhibiting significantly lower usage. There are also a few clusters exhibiting high usage; e.g. cluster 5 for the control group and cluster 9 for the treatment group.

4. METHODS AND RESULTS

The analyses described below, aim to provide answers to the two objectives outlined in Section 1. In our first analysis, we estimate the average treatment effect of the recommendation algorithm on EoC scores, using a simple linear regression model, with the following two categorical variables and the interaction between them; (i) the first categorical variable TC, comprises of two levels: the first represents the Treatment group that watched the personalized recommended videos and took the corresponding mini-assessments, and the second level corresponds to the Control group; (ii) the second categorical variable Previous Achievement Level, comprises of five categories, each corresponding to a different level of achievement in the Pre-test. Level 1 stands for the lowest achievement, whereas the highest level is coded by level 5. Then, the linear regression model with the above two predictors and their interaction is given by:

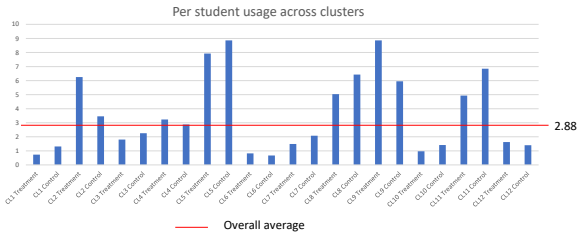


Figure 3: Average usage per student for different clusters, for both treatment and control groups

$$y = \mu + \beta_1\ TC + \beta_2\ Achievement + \beta_3\ (TC \times Achievement) + \epsilon \quad (1)$$

where y represents the EoC score and we further assume that $\epsilon \sim N(0, \sigma^2)$. Based on this model, the estimate of the average treatment effect of the personalized recommendation on EoC score, is the coefficient β_1 corresponding to the variable TC. Further, estimates of standard errors of the regression coefficients are based on cluster-robust estimators [2]. To answer the first research question discussed in Section 1, we test $H_0^{TC} : \beta_1 = 0$ vs. $H_1^{TC} : \beta_1 \neq 0$. The coefficient β_1 is the difference between the mean EoC score of the Treatment and the Control group, after accounting for the effect of all the other covariates. The estimated coefficients (scaled) and corresponding p-values are reported in Table 2. Table 2 shows that the achievement levels are statistically significant, while the treatment effect (i.e., the impact of the developed recommendation algorithm) is not. Further, there is a small positive significant effect for the interaction of the treatment with Achievement level 2. However, as shown in Figure 3, usage patterns vary widely across different groups (clusters) of students.

To that end, and in order to gain a deeper understanding of how the average treatment effect behaves across different IVLE usage levels, we fit model (1) separately on groups of students exhibiting different usage levels. After some initial exploratory analysis, we divided the students in approximately evenly distributed usage groups as shown in Table 3. The results are summarized in Table 3, whose first column specifies the usage levels of the group. As an example, students who have taken at least 10 mini-assessments tests, are categorized as a group with usage level 10 or higher.

Table 2: Estimated coefficients and corresponding p-values for Model (1)

Variable	Scaled Coefficient	p-value
Intercept	348.45	<0.001
Treatment	-0.99	0.32
Achievement level 2	11.28	<0.001
Achievement level 3	23.14	<0.001
Achievement level 4	35.31	<0.001
Achievement level 5	50.02	<0.001
TC*Achievement level 2	1.94	0.05
TC*Achievement level 3	0.94	0.34
TC*Achievement level 4	1.25	0.21
TC*Achievement level 5	1.09	0.27

Note: The scaled coefficients are obtained by dividing the estimated coefficients by their standard error.

The second and third columns contains= the p-values corresponding to the test $\beta_1 = 0$ and the scaled version of the estimated coefficients, respectively. As it is evident from Table 3, the first few rows that correspond to lower usage groups, have high p-values and thus the average treatment effect is not statistically significant. The treatment effect becomes significant for students who used the platform more extensively (≥ 48).

The model also controls for the level of achievement of students. Table 4 presents the results for the β_2 regression coefficient for different usage levels. The corresponding p-values are given in parentheses. It can be seen that the effect is statistically significant (marked in bold font) across almost all Previous Achievement levels and usage levels, as expected based on the overall results presented in Table 2. Further, this result is in accordance with a large body of literature that has found a positive association between level of math preparation and test scores (see, e.g.,[16, 15] and references therein). Further, the magnitude of the coefficient is larger for higher achievement levels.

Model (1) also estimates the interaction effect between the treatment and the Previous Achievement level. Table 5 summarizes the scaled estimates of the interaction effects and the p-values (given in parentheses). Since the Previous Achievement level has 5 categories, we obtain the estimates for all the levels except the baseline category, i.e., Previous Achievement level 1, which is absorbed in the intercept of the model. As usage increases, Table 5 displays more significant interaction effects (in bold font) between treatment and achievement level as compared to low usage groups. Note that due to lack of data in selected categories, some of the interaction effects could not be estimated and hence left blank.

Note that most of the interaction effects are not statistically significant. There are selected ones with a positive coefficient, corresponding to higher achievement levels (3 and above) for high usage groups (e.g., 33 and 65). Analogously, there are selected interaction effects with a negative coefficient corresponding to the lower achievement level 2, and relative high usage level.

To answer the second research question on the relationship between the performance of the students in mini-assessments and in the EoC test, we obtain the *Average Mini-Assessments*

Table 3: Usage-wise effect of the recommendation: p-values and scaled coefficients for different usage levels

Usage Level	p-value	Scaled Coefficient	Sample size
9	0.64	0.46	1097
13	0.40	0.85	932
27	0.29	1.07	515
33	0.17	1.39	411
48	0.02	2.41	254
52	0.01	2.56	230
55	0.05	1.93	207
59	0.06	1.91	183
65	0.02	2.31	140
74	0.08	1.78	92

Table 4: Usage-wise effect of the Previous Achievement level: p-values and scaled coefficients for different usage levels

Usage Level	Level 2	Level 3	Level 4	Level 5
9	4.65 (**<0.001**)	7.41 (**<0.001**)	10.67 (**<0.001**)	15.76 (**<0.001**)
13	4.94 (**<0.001**)	7.51 (**<0.001**)	- 10.74 (**<0.001**)	15.37 (**<0.001**)
27	1.87 (0.06)	2.64 (**0.008**)	4.61 (**<0.001**)	6.76 (**<0.001**)
33	2.45 (**0.01**)	3.02 (**0.002**)	4.72 (**<0.001**)	6.19 (**<0.001**)
48	3.23 (**0.001**)	3.19 (**0.001**)	4.18 (**<0.001**)	5.41 (**<0.001**)
52	3.16 (**0.002**)	3.29 ((**0.001**))	4.16 (**<0.001**)	5.42 (**<0.001**)
55	2.97 (**0.003**)	3.07 (**0.002**)	3.97 ((**<0.001**))	5.41 (**<0.001**)
59	2.05 (**0.04**)	1.92 (**0.05**)	2.65 (**0.008**)	3.58 (**<0.001**)
65	-	-0.13 (0.89)	1.83 (0.06)	4.15 (**<0.001**)
74	-	0.50 (0.62)	1.34 (0.18)	2.70 (**0.008**)

Table 5: Usage-wise interaction effect of treatment and Previous Achievement level: scaled coefficients (p-values) for different usage levels

Usage Level	TC * Level 2	TC * Level 3	TC * Level 4	TC * Level 5
9	-0.47 (0.64)	-0.20 (0.84)	-0.40 (0.69)	-0.06 (0.94)
13	-1.08 (0.27)	-0.68 (0.49)	- 0.81 (0.42)	-0.34 (0.74)
27	0.62 (0.54)	1.19 (0.23)	1.05 (0.29)	1.60 (0.11)
33	0.78 (0.43)	1.47 (0.14)	1.32 (0.19)	2.13 (0.03)
48	-2.82 (**0.005**)	-1.21 (0.22)	-2.05 (**0.04**)	-
52	-2.85 (**0.004**)	-1.47 (0.14)	-2.13 (**0.03**)	-
55	-2.49 (**0.01**)	-1.18 (0.24)	-1.73 (0.08)	-
59	-2.40 (**0.02**)	-0.71 (0.48)	-1.68 (0.09)	-
65	-	2.56 (**0.01**)	2.01 (**0.04**)	2.85 (**0.005**)
74	-	-0.98 (0.32)	-1.28 (0.20)	-

Score for each of the $\sim$ 11,000 registered students, wherein the average is computed over the mini-scores for all the mini-assessments the student has completed.

Then, the following Analysis of Covariance model is fitted to the data. To control for the students math preparedness and school characteristics, we include the cluster information as a factor in the model.

$$y_{ij} = \mu + \alpha_i + \beta x_{ij} + \varepsilon_{ij}, \quad (2)$$

wherein y_{ij} is the EoC score and x_{ij} is the Average Mini-Assessments Score for the j^{th} student in the i^{th} cluster. Further, μ is the overall mean effect and α_i is the additional effect due to the assignment of the student to the $i-$th cluster that accounts for prior math knowledge, grade and school characteristics of the students.

Table 6 depicts the estimated regression coefficients, their standard errors, together with the value of the test statistic and the p-value corresponding to the significance test for each of the coefficients. All p-values are significantly smaller than the nominal 0.05 (or 0.01) level, thus indicating that the corresponding predictor has a significant effect on the EoC test score. The estimated coefficient for the mini-assessment is 1.15. This small, but statistically significant coefficient indicates that an increase of one point in the average student score on the mini-assessment corresponds to an expected improvement in the EoC score of 1.15 (the corresponding scaled regression coefficient is 7.46) points. At first glance, this relationship between the average mini-score performance and the EoC test seems of limited practical significance. However, when examining the distribution of EoC scores across all students ($\sim 90,000$) that used the Math Nation platform at some point in time (not necessarily participants in the current study), we find that about 1.9% are within 1 point of the passing threshold. Hence, in light of this information, it is reasonable to posit that the recommendation algorithm would have been beneficial for a good number of students, if it were adopted and used by all platform participants.

Table 6: Results of the Analysis of Covariance model: Response EoC Score; categorical predictor cluster and numerical predictor Average Mini-Assessments Score

Coefficients	Estimate	Std. Error	t-value	p-value
Intercept	464.29	2.61	178.18	<2e-16
Cluster 3	29.79	3.63	8.21	3.9e-16
Cluster 4	29.61	2.77	10.70	<2e-16
Cluster 5	37.06	2.92	12.70	<2e-16
Cluster 6	38.99	3.13	12.46	<2e-16
Cluster 7	36.84	2.77	13.29	<2e-16
Cluster 8	49.74	2.92	17.02	<2e-16
Cluster 9	53.47	2.87	18.62	<2e-16
Cluster 10	73.48	2.90	25.33	<2e-16
Cluster 11	68.89	3.21	21.49	<2e-16
Cluster 12	75.34	2.88	26.13	<2e-16
Avg. Mini-Assessments	1.15	0.15	7.46	1.3e-13

5. DISCUSSION

The analysis of the data from the randomized control study provide a number of useful insights in designing recommendation strategies for IVLE. Firstly, the recommendation algorithm holds a lot of promise, but as it is well known in reinforcement learning, it requires adequate amount of usage to "explore" various possibilities in order to maximize expected reward. The adequate usage requirement is also discussed in the literature evaluating recommendation strategies for Massive Online Open Courses; see [6, 17, 18] and references therein. As mentioned in Section 3, an initial evaluation of the proposed algorithm during its development phase based on synthetic data indicated that it starts yielding satisfactory results, in terms of students improving their performance on the mini-assessments, once students follow its recommendations for over 45 times. The results of the analysis in Section 4 are in line with the aforementioned finding. As Table 3 indicates, the recommendation strategy shows significant impact starting from a usage level of 48. Further, note that in our study the primary outcome under consideration is the EoC test that takes place at the end of the academic year, as opposed to a more direct outcome related to the recommendation algorithm, such as performance over time on the mini-assessment tests. In many studies in the literature (e.g., [1, 22], assessment of a recommendation algorithm was based on more immediate outcomes (e.g., the mini-assessments in our setting), as opposed to a more distal outcome, such as the EoC. Nevertheless, the results of our experiment indicate that with stronger student engagement the developed algorithm could be more widely beneficial.

To address the issue of low usage, a new experiment has been designed, wherein the teachers are directly involved in the implementation of the recommendation system in the classroom, which is expected to yield higher levels of engagement of students with the IVLE platform. This experiment is under way at the time of this publication.

It is also worth mentioning that our first analysis was of "Intent-to-Treat" type, because it evaluated the effect of being randomly assigned to treatment or control groups without consideration of the extent that students used the recommendation strategy. On the contrary, traditional Complier Average Causal Effect analysis [21, 25] is based on "Treatment-on-the-Treated" principle, wherein one estimates the treatment effect for those who complied with the treatment. The latter constitutes a direction of future research.

Another issue of broader interest is that many IVLE recommendation algorithms are designed to assign test problems in an adaptive way, as opposed to assigning videos that Math Nation does. However, in the modified implementation of the algorithm currently under evaluation, the student can skip watching the recommended video and take the mini-assessment directly; in case, (s)he gets less than two of the questions correctly, the algorithm recommends to watch the segment of the video that covers the corresponding material and then retake the mini-assessment. This modification aims to enhance the emphasis of the recommendation algorithm on solving problems, but at the same time enable students to review relevant material to questions that they answered incorrectly.

6. ACKNOWLEDGMENTS

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305C160004 to the University of Florida. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

7. REFERENCES

[1] J. Bassen, B. Balaji, M. Schaarschmidt, C. Thille, J. Painter, D. Zimmaro, A. Games, E. Fast, and J. C. Mitchell. Reinforcement learning for the adaptive scheduling of educational activities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020.

[2] A. C. Cameron and D. L. Miller. A practitioner's guide to cluster-robust inference. *Journal of human resources*, 50(2):317–372, 2015.

[3] Y. Chen, X. Li, J. Liu, and Z. Ying. Recommendation system for adaptive learning. *Applied psychological measurement*, 42(1):24–41, 2018.

[4] M. Chi, K. VanLehn, D. Litman, and P. Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1-2):137–180, 2011.

[5] S. Doroudi, K. Holstein, V. Aleven, and E. Brunskill. Towards understanding how to leverage sense-making, induction and refinement, and fluency to improve robust learning. *International Educational Data Mining Society*, 2015.

[6] K. S. Hone and G. R. El Said. Exploring the factors affecting mooc retention: A survey study. *Computers & Education*, 98:157–168, 2016.

[7] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[8] K. R. Koedinger, J. R. Anderson, W. H. Hadley, and M. A. Mark. Intelligent tutoring goes to school in the big city. 1997.

[9] K. R. Koedinger, E. Brunskill, R. S. Baker, E. A. McLaughlin, and J. Stamper. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3):27–41, 2013.

[10] J. A. Kulik and J. Fletcher. Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of educational research*, 86(1):42–78, 2016.

[11] W. Ma, O. O. Adesope, J. C. Nesbit, and Q. Liu. Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of educational psychology*, 106(4):901, 2014.

[12] S. Mojarad, A. Essa, S. Mojarad, and R. S. J. de Baker. Studying adaptive learning efficacy using propensity score matching. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge (LAK18)*, 2018.

[13] P. Morgan and S. Ritter. An experimental study of the effects of cognitive tutor algebra I on student knowledge and attitude. Pittsburgh, CA: Carnegie Learning Inc., 2002.

[14] R. Murphy, L. Gallagher, A. Krumm, J. Mislevy, and A. Hafter. Research on the use of Khan Academy in schools. Menlo Park, CA: SRI Education, 2014.

[15] S. A. Niaki, C. P. George, G. Michailidis, and C. R. Beal. The impact of an online tutoring program for algebra readiness on mathematics achievements; results of a randomized experiment. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, pages 363–372. ACM, 2019.

[16] J. F. Pane, B. A. Griffin, D. F. McCaffrey, and R. Karam. Effectiveness of cognitive tutor algebra I at scale. *Educational Evaluation and Policy Analysis*, 36(2):127–144, 2014.

[17] J. Reich. Mooc completion and retention in the context of student intent. *EDUCAUSE Review Online*, 8, 2014.

[18] J. Reich and J. A. Ruipérez-Valiente. The mooc pivot. *Science*, 363(6423):130–131, 2019.

[19] S. Ritter, J. Kulikowich, P. Lei, C. McGuire, and P. Morgan. Big data comes to school: Implications for learning, assessment, and research. In *15th International Conference on Computers in Education: Supporting Learning Flow through Integrative Technologies, ICCE 2007*, pages 13–20, 2007.

[20] J. P. Rowe and J. C. Lester. Improving student problem solving in narrative-centered learning environments: A modular reinforcement learning framework. In *International Conference on Artificial Intelligence in Education*, pages 419–428. Springer, 2015.

[21] B. J. Sagarin, S. G. West, A. Ratnikov, W. K. Homan, T. D. Ritchie, and E. J. Hansen. Treatment noncompliance in randomized experiments: Statistical approaches and design issues. *Psychological methods*, 19(3):317, 2014.

[22] S. Shen, M. S. Ausin, B. Mostafavi, and M. Chi. Improving learning & reducing time: A constrained action-based reinforcement learning approach. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 43–51, 2018.

[23] S. Shen, B. Mostafavi, C. Lynch, T. Barnes, and M. Chi. Empirically evaluating the effectiveness of pomdp vs. mdp towards the pedagogical strategies induction. In *International Conference on Artificial Intelligence in Education*, pages 327–331. Springer, 2018.

[24] S. Steenbergen-Hu and H. Cooper. A meta-analysis of the effectiveness of intelligent tutoring systems on college students' academic learning. *Journal of Educational Psychology*, 106(2):331, 2014.

[25] E. A. Stuart, D. F. Perry, H.-N. Le, and N. S. Ialongo. Estimating intervention effects of prevention programs: Accounting for noncompliance. *Prevention Science*, 9(4):288–298, 2008.

[26] K. Vanlehn. The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265, 2006.

[27] K. VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221, 2011.

[28] L. S. Vygotsky. *Mind in society: The development of higher psychological processes*. Harvard university press, 1980.

Student Practice Sessions Modeled as ICAP Activity Silos

Adam M. Gaweda, Collin F. Lynch
North Carolina State University
Raleigh, NC, USA
agaweda,cflynch@ncsu.edu

ABSTRACT

There are a number of novel exercise types that students can utilize while learning Computer Science, each with its own level of complexity and interaction as outlined by the ICAP Framework [10]. Some are *Interactive*, like solving coding problems; *Constructive*, like explaining code; *Active*, like retyping source code; and *Passive*, like reviewing slides. To date, there has been little research on how students vary their study and engagement habits by exercise type and when they do so. In this paper, we present our findings on student activity sequences from an online professional development course. We isolated student activities into sessions and then produced activity transition visualizations to compare the behavior of students who complete the course to those who do not. We then used multiple factor analyses to examine how students transition from one type of activity to the next. From this analysis we identified *platform silos* in student's work. We further expand this concept to the presence of *activity silos* grouping by type. We find that this siloing behavior is consistent in both completers and non-completers but is weaker for the latter group. Finally, we discuss our findings and how instructors and researchers may use this information to ensure that students show persistence through practice.

Keywords

novel exercises, ICAP framework, study sessions, activity sequences, platform silos, activity silos, student modeling

1. INTRODUCTION

CS Education have introduced a number of novel exercise types to better scaffold students' experiences. These include retyping source code [14], arranging scrambled code fragments (Parsons Puzzles) [21], debugging provided code [8], predicting output [26], fill in the blanks [4], self-explanation [4], and small scale coding exercises [2, 12]. Each of these exercise types can also be mapped onto the ICAP framework [10]. This framework defines four categories of instructional activities based upon students' level of engagement: Interactive, Constructive, Active, and Passive. *Passive learning* includes reading static course materials or watching lecture videos. *Active learning* is described as rehearsing or copying solution steps. *Constructive learning* includes self-explanation of content or creation of novel externalized outputs like summaries. Finally, *Interactive learning* involves directly engaging with a peer, agent, or instructor to explore information and receive feedback which can be expanded upon.

While these exercise types have made their way into classrooms there is little evidence of how these types of engagement interact with one-another. Traditional intervention studies have focused on the overall impact of one or more exercise types [20, 19, 14, 8], or on the automated selection/recommendation of future excercises based upon a student model [27], but not on how students work with or across them in the absence of guidance. Nor has this recommendation work been extended to nontraditional learning contexts. Absent an understanding of how students orchestrate multiple interaction modes we face challenges in scaffolding effective learning opportunities and in evaluating the impact of novel learning environments. Providing students with ineffective, or overly complex learning opportunities risks trapping them in a fail/skip practice cycle that would inhibit any functional learning gains [17].

In this paper we report our investigation of how students direct their practice of CS concepts when presented with a set of options. Our study was conducted in the context of an online professional development course for Python programming. This course is part of a research study funded by the Department of Labor to create novel learning pathways for existing technical professionals to move into AI and Data-Science areas. We extracted students' practice/study sessions and analyzed the activity transitions within each session. We then analyzed these activity sequences to answer the following research questions (RQs):

RQ1 Can we replicate the existence of *platform silos* introduced in [1] with a new dataset?
RQ2 Are there common activity transitions between students?
RQ3 How do activity sequences connect with the ICAP Framework?
RQ4 How do the practice sessions of completers of the course differ from non-completers?

To answer these questions, we first produced and analyzed a set of activity transition diagrams for students in the course comparing those who completed the course to those who did not. Through this analysis, we confirmed the presence of *platform silos* which we extend this notion to include *activity silos*, where students primarily focus on a single mode of engagement (consistent with ICAP) during a given practice session. When students did transition between modes, it was only to move up the ICAP chain and never to 'downgrade' to a lower level of engagement. We support our findings through two different factor analyses, which help explain the 42-62% of variance between the sessions. From

Adam Gaweda and Collin Lynch "Student Practice Sessions Modeled as ICAP Activity Silos". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 595-601. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

our results, we found ICAP categories isolated into individual sessions, as well as LMS content consumption and quiz taking.

2. BACKGROUND

2.1 ICAP Framework

The ICAP Framework seeks to classify the modes of student engagement while engaging in learning activities [10]. These categories, *Interactive, Constructive, Active* and *Passive* respectivel. *Passive* engagement includes activities such as reading a text or observing a video. *Active* engagement includes rehearsing steps or copying solutions. *Constructive* engagement includes self-explanation or comparing and contrasting materials. Finally, *Interactive* engagement includes responding and interacting with an agent, system, or another person. Tthe framework is hierarchical, suggesting I > C > A > P, or that activities with higher levels of engagement promote the greatest levels of learning.

Chi and Wylie present a literature review of several empirical studies supporting their ICAP hypothesis [10]. The first study consisted of all four modes of engagement in materials science and showed learning improved significantly at a rate of 8-10% per mode. They then present two studies which used active, passive, and constructive modes in evolutionary biology and plate tectonics. Finally, Chi and Wylie present comparisons of two modes across note taking, concept mapping, and self-explanation. In each of these studies, the results again showed that higher modes of engagement had higher learning gains.

Chi and Wylie's work, as well as our own personal communications with Chi [9], note that identifying the mode of a particular activity can be a non-trivial task. For example, a toy example task could be presenting steps toward making a peanut butter and jelly sandwich given randomly shuffled segments of instructions. This task could be constructed as an *active* exercise if the student already knows the recipe and the task is simply picking the appropriate sequence from the list of steps. However, if the student had not learned the appropriate order, then the task of figuring out the right sequence would be construed as a *constructive* exercise. Computer Science has a similar task, known as Parsons Puzzles, that mirrors this toy example, discussed in more detail in the following section.

2.2 Novel Exercise Types

In this section, we describe eight different exercise type studied in Computer Science education, provide some research background on the exercise type, and justifications for which ICAP mode we will classify them as for of our study.

2.2.1 Typing Exercises

Typing Exercises (TE) require students to retype source code that has been presented to them [14]. Typing Exercises can be used as *active* learning activities under the ICAP Framework as they require students to retype verbatim the code presented to them. Previously, we presented images of source code and showed that self-selected students that completed optional typing exercises earned higher course grades and submitted less code with build failures. Leinonen et. al. also presented optional typing exercises to students before programming tasks [19], but were not able to find the same results as ours. However their study only lasted two weeks and many of their selected participants did not attempt the exercises at all.

2.2.2 Fill in the Blank

Fill in the Blank ($FitB$) exercises remove a small portion of code from a snippet and asks students to 'fill in' the blank. Students need to have an understanding of the snippet as a whole to deduce what needs to be included at a particular blank location. Reviewing incomplete worked examples reduces ineffective self-explanations and enhances the transfer of learned materials [4, 5]. Based on the results from Atkinson et. al., we consider FitB-style exercises to require lower levels of engagement than self-explanation, and thus classify them as an *active* ICAP mode.

2.2.3 Parsons Puzzles

Parsons Puzzles (PP) present snippets of code that have been separated into segments and then shuffled in order [21]. Students are then tasked with placing the segments back into the correct order. While Parsons Puzzles are helpful for learning how to structure code, performance on Parsons Puzzles has not been shown to correlate with students' ability to read or trace code [11, 20] and 'distractor' variants are not beneficial to young learners [13, 16]. These findings further support our research goals, as not every exercise type may be beneficial for learning all the technical skills necessary for Computer Science.

Chi and Wylie's definition of constructive modes of engagement include "learners generate or produce additional externalized outputs... beyond what is provided". As mentioned in the previous section, Parsons Puzzles are similar to our toy peanut butter and jelly exercise. While Parsons Puzzles could be construed as *active*, students may not fully comprehend the appropriate order of code syntax and must figure out the right sequence as part of the exercise. In our communications with Chi about Parsons Puzzles, Chi states that determining the particular ICAP mode for an activity can be non-trivial [9]. 'If a student already [knows] the recipe, then re-ordering it is just picking out the sequencing, guided by the sequence information [they] already know' then it is *active*. However, 'if the student [has] not learned the order from some other source, and you are asking her to figure out the right sequence', it is a *constructive* exercise. Since novices may not have proficiency at the time of the exercise, we elect to use the upper bound ICAP mode and consider Parsons Puzzles as a *constructive* learning activity.

2.2.4 Output Prediction

Output Prediction (OP), also known as variable tracing, exercises ask students to analyze code and then state the expected outputs of code execution or the expected value of a variable as the code progresses. Often, variable tracing is done during conditional and loop instruction to demonstrate how the values of the variables change after each iteration. Like Parsons Puzzles, OP-style exercises require students to process code snippets and externalize their expected outputs. Thus, we consider output prediction as another *constructive* learning activity.

2.2.5 Self-Explanation

Self-explanation (SE) exercises present students with source code and ask the student to explain how the code operates, describe the overall efficiency of the code, or create a documentation string to appear as a comment for the program or function. These are open-ended exercises that are subjective in nature and are considered to be *constructive* [10]. However, Chi and Wylie do note that students' may treat the self-explanation activity as *active* if "the student's self-explanation is verbatim to what was read". However, novices may struggle with reading and evaluating programming code in a linear fashion, focusing more on what each line of code did, rather than how each line interacted with each other, or in general produce poor explanations [25, 5]. While *constructive* SE activities may produce higher learning gains than lower-level modes, they may also not be the most appropriate activity for students who are struggling, Thus, similar to our

decisions for Parsons Puzzles, we use the upper bound to classify self-explanation as a *constructive* learning activity.

2.2.6 Find and Fix the Bug

Resolving errors, or debugging, is one of the first hurdles students encounter when learning to program [3]. Once they find that an error has occurred, it will be necessary for them to resolve it before addressing any remaining subgoals for their solution. For the purposes of our study, we separated debugging into two separate activities - Find the Bug (FnB) and Fix the Bug (FxB). Find the Bug exercises present students with code that contains a common misconception for novices. Instead of resolving the error, students are asked to highlight the area of code where this error exists. Though the ICAP framework considers highlighting text as an *active* learning activity, Chi and Wylie define *constructive* behaviors as requiring some level of "inference", or adding in additional detail or qualification. Since students must assess if a line of code is 'correct' or not, they are producing qualifications and therefore, we elect to label FnB exercises as *constructive*.

Fix the Bug exercises follow the natural progression of debugging tasks by requiring students to resolve broken code[8]. FxB activities could be considered *constructive* or *interactive* depending on the context. Similar to FnB exercises, Chi and Wylie include 'repairing' as a *constructive* behavior. However, FxB-style exercises can also be *interactive* because students often rely on the code interpreter's feedback during the debugging process. Fixing one error may produce new errors with new feedback, or the repair made by the student could be incorrect. Thus, we again choose use the upper bound to label FxB exercises as *interactive*.

2.2.7 Coding Exercises

The final exercise type we used in our study is the de facto standard of introductory CS courses - the Coding Exercise (CE). While Computer Science is more than programming, coding exercises are often used by instructors as graded course material for students to demonstrate their understanding of the current course topic. There has been work on the use of 'many-small programs' and 'simple syntax exercises', which simply require students to complete small-scale coding exercises to become familiar with the a particular implementation before utilizing it as part of larger-scale problems [2, 12]. In both cases, completing these smaller-scale programming exercises improved student performance and yielded happier students. Students often rely on feedback from the interpreter as they construct their solutions and more than likely need to debug their own work during this process. Thus, we consider Coding Exercises as an *interactive* learning activity.

2.3 Student Modeling and Activity Mining

Seshadri et. al. analyzed how student study sessions operated across multiple platforms for three separate courses [1]. Their results found that given multiple education platforms, students will often operate within *platform silos*, or only utilize one educational platform during an individual study session. Of the student sessions, more than 90% of them included only one platform. In a follow-up study, they compared the activities of higher performing students to the lower performing group and showed that both groups were most likely to stick within platform silos [15]. Their work serves as the motivation for our RQ1. Our hypothesis is that the presence of these 'platform silos' will continue to hold across other educational platforms not studied in their research.

3. STUDY

3.1 Design

We studied problem solving in the context of an online professional development course in Python programming. This is a preparatory course for a series in AI that is aimed at non-traditional students making a career transition. The course used the Moodle Learning Management System and TYPOS, a CS exercise platform [14], and is organized into 10 modules Each module includes a set of static reading material, lecture slides, prerecorded videos, optional practice exercises, and a module assessment. There were 24 optional exercises per module, 3 exercises for each of the 8 types previously described. Students were free to work on the practice exercises or assessments as much as they liked. The only requirement for progressing to the next module was to earn a passing grade (80% or higher) on the prior module's assessment. In order to complete the course, students needed to earn passing grades on all assessments.

We had 69 students consent to the study. Of those students, 37 successfully completed the course. Student interactions on both platforms were logged. We omitted some Moodle interactions such as like "file downloading" and "viewing the course" which did not pertain to the explicit learning actions we were foucused on.

The resulting dataset contained a total of 29,190 interactions from all the students. We then used a similar strategy as [1] to extract user sessions from these interactions based upon an exploratory analysis of the gaps between interactions. The time deltas between course interactions were measured. If a delta between interactions exceeded a predefined cutoff threshold, that session was considered over, and a new session was created. This was repeated for all interactions a student had for the course. While Seshadri et. al. used a 40 minute cutoff to establish the end and start of a new session, we chose a 60 minutes as it was our most frequently observed delta between interactions. We extracted 1,313 sessions in total. Students that completed the course accounted for 71%, or 20,748, of the course interactions, with 1,041 sessions total, at an average of 28.1 ($\pm$17.9) sessions per completer.

3.2 Activity Session Transition Probabilities

The route that students take through online materials can be modeled as discrete Markov processes, in which each state represents an activity within the session. For example, a student may transition from reviewing lecture slides to viewing lecture videos on Moodle, or $MS \rightarrow MV$. Jeffries et. al. [17] used a similar process to analyze success and help seeking behaviors with students in an introductory CS course.

Figure 1 visualizes the transition probabilities for completers: transitions within TYPOS appear as dashed blue lines, transitions within Moodle are solid red lines, and transitions between TYPOS and Moodle are solid black lines. For visibility, only transitions involving the start/end of a session or those with a frequency above 5% are presented. Module assessment (MA) accounted for 39% of starting session behavior, TYPOS practice accounted for 36%, and lecture slides and videos (Content Consumption) accounted for 26%. This figure shows students' practice was largely siloed by platform with each session taking place within a single mode of interaction. With the exception of the $MS \rightarrow TE$ transition, students either interacted with TYPOS or Moodle, but rarely together. Since MA showed the highest starting session probability, one assumption is that students enrolled in the course with prior coding experience may have reviewed the course material to become familiar with Python syntax before going on to complete the module assessment.

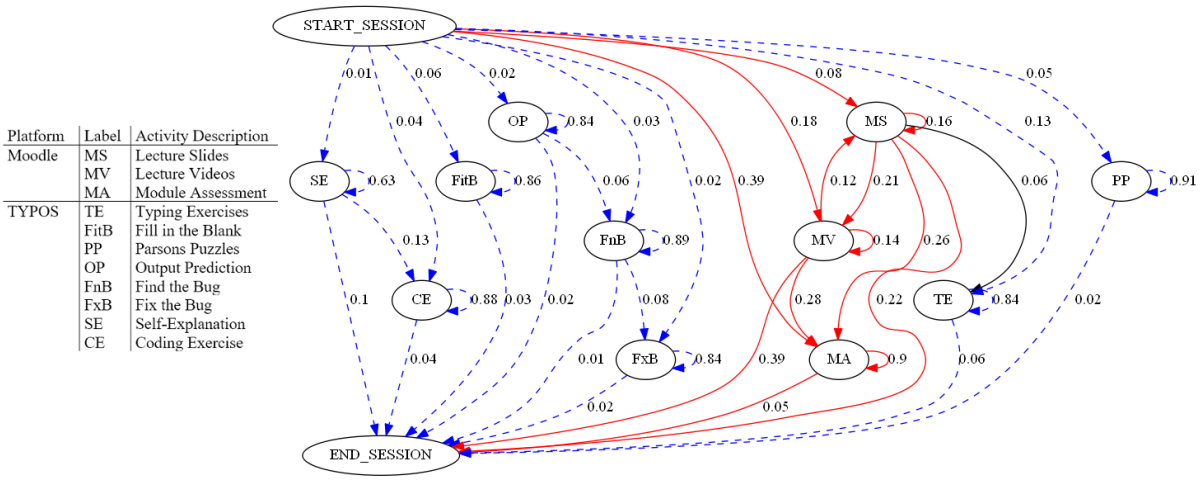


Figure 1: Completer activity transition probabilities during sessions.

One interesting observation from Figure 1 is that TYPOS practice sessions involved only one or a small subset of the exercise types available to the students. For example, the TE, PP, and $FitB$ exercises were typically completed alone. Among two exercise pairs, $SE \rightarrow CE$, $OP \rightarrow FnB$, and $FnB \rightarrow FxB$ showed higher probabilities than ending the session.

We can further classify the activities by their respective ICAP modes. For example, the $SE \rightarrow CE$ transition can also be viewed as a Constructive$\rightarrow$ Interactive transition, $MS \rightarrow TE$ can be viewed as Passive$\rightarrow$ Active, and so on. From this perspective, transitions between activities rarely 'downgraded' to a lower mode. With the exception of $MS \rightarrow MA$ and $MV \rightarrow MA$, mode changes primarily shifted by one level of engagement.

3.3 Activity Session Factor Analysis

The probabilities shown in Figure 1 raise new questions about students' practice behaviors. Not only were we able to confirm the existence of *platform silos* from our RQ1, but based on the observed transition probabilities, we found that students may also operate within what we term an *activity silo*, focusing primarily on one ICAP mode per session. In this section we report on two factor analyses which we use to identify the latent variables for each practice session in order to strengthen our claim.

Factor Analysis is a dimension reduction method to describe the variability of observed variables into, potentially, lower latent variables, or factors. To prepare our dataset for factor analysis, each activity was converted into a binary value, representing the presence or absence of the activity in the session. For example, if a session only involves passive content consumption, the resulting vector for the session would be [1,1,0,0,0,0,0,0,0,0,0], where the 1s represent the presence of lecture slides and videos and 0 represents the absence of all other activities.

In order to evaluate the appropriateness of our data for factor analysis, we use the Bartlett's and Kaiser-Meyer-Olkin tests. Bartlett's test compares our correlation matrix against an identify matrix to test whether our samples are from populations with equal variance. Our samples were statistically significant ($\chi^2 = 3360.05, p = 0.0$) and thus we can continue with our factor analysis. Kaiser-Meyer-Olkin checks the adequacy for our variables to determine the suitability of factor analysis. Our KMO score was 0.83, which again shows our dataset is adequate.

The next step for our analysis was to determine the appropriate number of factors. Table 1 shows the eigenvalues for each factor and their cumulative variance. Based on these results, we utilized two separate factor analyses. The first analysis uses 3-factors to correspond to the 3 eigenvalues greater than 1, as suggested by Kaiser [18]. The second analysis increases to 5-factors based on the variance extraction rule, which specifies a 0.7 threshold for eigenvalues [6, 24, 23].

Table 1: Eigenvalues for Factor Analysis of Completers

Factor	Eigenvalue	Cumulative Variance
1	3.860695	30.51%
2	1.390921	37.21%
3	1.170659	41.79%
4	0.916401	52.26%
5	0.810977	62.27%
6	0.665925	60.91%
7	0.649612	69.59%
8	0.486275	70.05%
9	0.446064	55.23%
10	0.391094	55.42%
11	0.211376	55.42%

Our next task was to identify load factor thresholds for our latent variables. While there is no universal standard for loading thresholds, the goal is to only observe variables that share a strong association with each other and is a non-trivial process [22]. For our paper, we will focus our attention to variables above a 0.50 (or 25% of the variable's variance) threshold, highlighting them in our tables as green. Since the difference between a 0.50 and 0.49 loading is minimal, we will also highlight values greater than 0.4 (or 16% variance) in yellow for additional reference.

Table 2 shows the factor load values for our 3-factor analysis. F1 has high loadings for OP, FnB, FxB, SE, and CE. If we consider the ICAP modes for this factor, this indicates a transition of *Constructive* $\rightarrow$ *Interactive* TYPOS Practice. F2 has high loadings for $FitB$ and PP, or *Active*$\rightarrow$ *Constructive* TYPOS Practice. Finally, F3 has high load for MS, or *Passive* Moodle Interaction. From Table 2, we once again can confirm the presence of *platform silos*, however we expand our factor analysis in order to see the presence of *activity silos*. Moreover, 3-factors only accounts for 41.79% of the cumulative variance and so using the 0.7 eigenvalue threshold will allow us to account for 62.27%.

And finally, Table 3 shows the factor load values for our 5-factor analysis. Similar to Table 2, we see a separation between

Table 2: Loadings for 3 Factor Analysis for Completers. Values greater than 0.4 are in yellow and greater than 0.5 are in green.

Activity	F1	F2	F3
MS	0.0306	-0.0294	0.5073
MV	-0.1006	-0.0999	0.3891
MA	-0.0269	-0.2862	0.2177
TE	0.0193	0.4336	-0.1345
FitB	0.4703	0.5471	-0.0286
PP	0.3300	0.6726	0.0392
OP	0.6257	0.3885	0.0216
FnB	0.8215	0.2149	0.0307
FxB	0.8399	0.1546	0.0059
SE	0.6802	0.1020	-0.0933
CE	0.5019	0.0135	-0.1306

TYPOS activity and Moodle activity. Further, the activities for each factor are confined to a single ICAP mode, or within one mode. F1 contains mostly *Constructive* activities (as well as FxB), F2 contains *Active*→ *Constructive* activities, F3 contains *Passive* activities, and F4 and F5 contain *Interactive* activities. In addition, F3 and F4 show a separation between *Passive* Moodle content consumption and *Interactive* assessment taking. Thus, from the results of our 5-factor analysis, we confirm the presence of *activity silos* within our students.

Table 3: Loadings for 5 Factor Analysis for Completers. Values greater than 0.4 are in yellow and greater than 0.5 are in green.

Activity	F1	F2	F3	F4	F5
MS	0.0224	-0.0065	0.2411	0.1061	-0.0150
MV	-0.0836	-0.1009	0.9838	-0.0982	-0.0264
MA	-0.0438	-0.1664	0.1175	0.9759	-0.0124
TE	0.0364	0.3448	-0.0843	-0.2002	-0.0106
FitB	0.4344	0.5629	-0.0451	-0.0445	0.0772
PP	0.2740	0.7467	0.0004	-0.0193	0.0369
OP	0.5520	0.4568	-0.0008	0.0282	0.1792
FnB	0.8419	0.2321	0.0035	-0.0110	0.0829
FxB	0.8759	0.1568	0.0058	-0.0255	0.0980
SE	0.5963	0.1562	-0.0318	-0.0450	0.2602
CE	0.3227	0.0549	-0.0631	-0.0088	0.8891

3.4 Comparing Completers to Non-Completers

Having shown the basic activity structures and identified relevant factors we then chose to explore was the difference between completer and non-completer students. We used the same methods for the non-completer group for comparison. There were 32 students that failed to complete our course. Non-completers made 8,442 course interactions across 341 sessions, with an average 10.7 (±7.8) sessions per non-completer.

We first produced the same transition probabilities diagram for non-completer activity sessions, seen in Figure 2. Similar to completers, module assessment accounted for 31% of starting session behavior, TYPOS practice accounted for 49%, and lecture slides and videos (Content Consumption) accounted for 22%. Non-completers primarily operated within a single platform, though there was more interactions between Moodle and TYPOS. For example, 12% of MV's transitions migrated to TYPOS exercises and 5% of SE transitions migrated to MA. While completer students separated $SE \rightarrow CE$ and $OP \rightarrow FnB \rightarrow FxB$ transitions, these two sequences were combined for non-completers. However, this could potentially be due to the size differences. Both populations had similar population sizes, but non-completers did not complete each module assessment and would not produce as many sessions.

We then carried out the same factor analyses for non-completers. The results of a Bartlett's test showed statistically significant differences ($\chi^2 = 997.8, p > 0.0001$) and our KMO score was also adequate for analysis (0.77). Similar to Table 1, we found support for 3- and 5-factor analysis, seen in Table 4. We note that a 6-factor analysis is also possible, but to mirror the factor analysis for completers, we elected not to pursue it.

Table 4: Eigenvalues for Factor Analysis of Non-completers

Factor	Eigenvalue	Cumulative Variance
1	3.480694	26.26%
2	1.503161	34.57%
3	1.286916	41.39%
4	0.888718	47.44%
5	0.821608	54.51%
6	0.719444	62.12%
7	0.657629	66.31%
8	0.520019	63.79%
9	0.499220	57.46%
10	0.375677	57.69%
11	0.246915	57.69%

Table 5 shows our 3-factor analysis for non-completers. The same activities having high loadings as F1 and F2 as the 3-factor analysis for completers (Table 2) and also show similar *platform silos*. Likewise, the ICAP mode considerations are similar for each factor. F1 shows *Constructive*→ *Interactive* behaviors, F2 shows *Active*→ *Constructive* behaviors, and F3 shows *Passive* Moodle interaction.

Table 5: Loadings for 3 Factor Analysis for Non-completers.

Activity	F1	F2	F3
MS	0.0362	0.0014	0.4167
MV	-0.0334	0.0106	0.6616
MA	-0.0136	-0.2524	0.2999
TE	0.0756	0.4637	-0.0719
FitB	0.3176	0.6024	-0.0202
PP	0.1082	0.7404	0.0221
OP	0.5310	0.3499	0.1105
FnB	0.5573	0.4837	-0.0910
FxB	0.6636	0.3369	-0.1218
SE	0.8012	0.0568	0.0573
CE	0.5900	0.0201	0.0056

Table 6 shows our 5-factor analysis for non-completers. Non-completers maintained the *Constructive*→ *Interactive* connection for F1 and F2 also maintains the *Active*→ *Constructive* connection. The remaining factors do differ, F3 separated the $FnB \rightarrow FxB$ exercises from F1 and F4 focuses primarily on TE. The absence of MA was expected since course progression requires passing module assessments. From our analysis, we conclude that non-completers still operated within *activity silos*.

Table 6: Loadings for 5 Factor Analysis for Non-completers.

Activity	F1	F2	F3	F4	F5
MS	0.0271	0.0234	0.0080	-0.0129	0.4040
MV	-0.0191	0.0204	-0.0417	0.0373	0.7123
MA	0.0049	-0.1705	-0.0558	-0.1637	0.2922
TE	0.0541	0.2612	0.0605	0.9570	-0.0716
FitB	0.2259	0.6333	0.1698	0.1275	-0.0557
PP	0.0224	0.6734	0.1463	0.1925	-0.0155
OP	0.4707	0.4584	0.1711	0.0080	0.0735
FnB	0.2516	0.4546	0.6255	0.0378	-0.0543
FxB	0.3618	0.2049	0.8444	0.0752	-0.0706
SE	0.8072	0.0883	0.2525	0.0740	0.0496
CE	0.6201	0.1006	0.1092	-0.0054	-0.0206

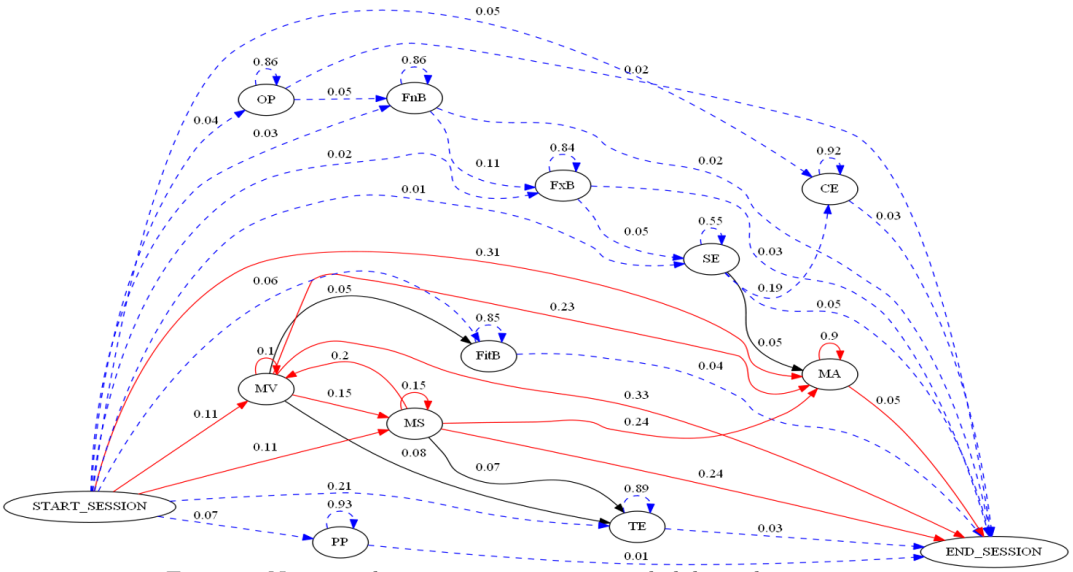


Figure 2: Non-completer activity transition probabilities during sessions.

4. DISCUSSION

The results from our probability transition diagrams confirm the presence of both *platform silos* and *activity silos* in student work. They also serve to highlight areas where educators and researchers can tailor more appropriate learning paths for students and in particular those students that may be struggling with course material.

Our study allowed students to self-select which activities they wanted to focus their attentions on. While this style of course design could be adopted, it does still present limitations. However, students that primarily focus on lower-level ICAP mode activities may be reluctant to move into high-level modes. Instructors or systems that can identify this stagnate practice behavior could encourage students to move into high-level ICAP modes. There is growing interest in the concept of *nudge theory* to "alter behavior without incentives or banning alternatives" [7] to encourage progression.

Similarly, we presented students with a number of different activities, at different complexities, for their learning experience. Based on our results, students were more than willing to complete each type of exercise. Some students even asked for more activities in our post-course survey. While increasing the workload for students and learning material creators, many of the activities we used are not overly complex and required a minimal amount of time to create, or from the students' perspectives complete. Activities like typing exercises or Parsons puzzles can be created from existing course materials and offer little incentive for students to cheat. They simply allow students an opportunity to practice the concepts they learned rather passively given to them, refining their understanding, before needing to apply it to problem solving activities like coding exercises.

5. LIMITATIONS

We acknowledge some limitations with our study. First, our course ran during the COVID-19 pandemic, which has altered many individuals' habits. Our population also contained non-traditional students who were balancing their studies with working from home and supporting other family members. Thus, non-completion may have been driven by external constraints that are not reflected in our dataset, and the observed habits may change somewhat during non-COVID times.

Second, the exercise types were presented in a consistent manner for each module. Thus they were implicitly sequenced with lower-level ICAP modes appearing on the top. As we mentioned in our introduction, discerning the appropriate order for 11 different activities is a non-trivial matter and measuring the appropriate order of exercise types was not a part of our study. Thus, we presented exercises in an order that progressively increased the level of engagement. This may have influenced next practice selections by students.

Finally, we acknowledge that the ICAP modes associated with each exercise type are somewhat subjective and open for discussion. Moreover the exact evaluation of exercises like Parsons Puzzles or self-explanation may require additional research and context. For the purposes of this study, when faced with uncertainty we classified exercises according to a higher level mode of interaction.

6. CONCLUSIONS

In this work, we extracted the practice and study session behaviors from non-traditional students learning Python. Among completers and non-completers of the course, they primarily focused on a single platform. The activities within these platforms were mapped to the ICAP framework. Further, we used factor analyses to identify the presence of *activity silos* within practice sessions. Completers and non-completers shared similar behaviors during these practice sessions, primarily focusing on one or two modes of engagement and rarely 'downgraded' to lower level modes.

We can utilize these activity sequences to help shape our overall course designs for ensuring student learning. Lower-level activities can provide students with the foundational knowledge necessary as a part of the technical skills for the content, while higher-level activities can refine and encourage additional learning gains. As the research in this area expands, we hope the information presented in this study encourages educators and researchers alike to provide practice in both levels and can serve as a guide for recommendations on how to best build long-term proficiencies.

Acknowledgements

This work was supported in part by the National Science Foundation under Grant DRL 1721160. Kristy Boyer, Eric Wiebe, and Collin F. Lynch (co-PIs).

7. REFERENCES

[1] Sheshadri Adithya, Niki Gitinabard, Collin F Lynch, Tiffany Barnes, and Sarah Heckman. Predicting student performance based on online study habits: A study of blended courses. *International Educational Data Mining Society*, 2018.

[2] Joe Michael Allen, Frank Vahid, Alex Edgcomb, Kelly Downey, and Kris Miller. An analysis of using many small programs in cs1. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 585–591, 2019.

[3] A. Altadmri and N. Brown. 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 522–527, New York, NY, USA, 2015. ACM.

[4] Robert K Atkinson, Sharon J Derry, Alexander Renkl, and Donald Wortham. Learning from examples: Instructional principles from the worked examples research. *Review of educational research*, 70(2):181–214, 2000.

[5] Robert K Atkinson and Alexander Renkl. Interactive example-based learning environments: Using interactive elements to encourage effective processing of worked examples. *Educational Psychology Review*, 19(3):375–386, 2007.

[6] Deborah L Bandalos and Sara J Finney. Exploratory and confirmatory. *The reviewer's guide to quantitative methods in the social sciences*, 93, 2010.

[7] Chris Brown. Digital nudges for encouraging developer actions. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 202–205. IEEE, 2019.

[8] Nick Cheng and Brian Harrington. The Code Mangler: Evaluating coding ability without writing any code. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 123–128, 2017.

[9] Michelene TH Chi. Private Communication, August 2020.

[10] Michelene TH Chi and Ruth Wylie. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist*, 49(4):219–243, 2014.

[11] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research*, pages 113–124. ACM, 2008.

[12] John Edwards, Joseph Ditton, Dragan Trninic, Hillary Swanson, Shelsey Sullivan, and Chad Mano. Syntax exercises in CS1. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, pages 216–226, 2020.

[13] Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, pages 20–29, 2017.

[14] Adam M Gaweda, Collin F Lynch, Nathan Seamon, Gabriel Silva de Oliveira, and Alay Deliwa. Typing exercises as interactive worked examples for deliberate practice in CS courses. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, pages 105–113, 2020.

[15] Niki Gitinabard, Tiffany Barnes, Sarah Heckman, and Collin F. Lynch. What will you do next? A sequence analysis on the student transitions between online platforms in blended courses. In *Proceedings of the 12th International Conference on Educational Data Mining, EDM 2019, Montréal, Canada, July 2-5, 2019*, 2019.

[16] Kyle James Harms, Jason Chen, and Caitlin L Kelleher. Distractors in parsons problems decrease learning efficiency for young novice programmers. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, pages 241–250, 2016.

[17] Bryn Jeffries, Timothy Baldwin, Marion Zalk, and Ben Taylor. Online tutoring to support programming exercises. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, pages 56–65, 2020.

[18] Henry F Kaiser. The application of electronic computers to factor analysis. *Educational and psychological measurement*, 20(1):141–151, 1960.

[19] Antti Leinonen, Henrik Nygren, Nea Pirttinen, Arto Hellas, and Juho Leinonen. Exploring the applicability of simple syntax writing practice for learning programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 84–90. ACM, 2019.

[20] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the fourth international workshop on computing education research*, pages 101–112. ACM, 2008.

[21] Dale Parsons and Patricia Haden. Parson's programming puzzles: A fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ACE '06, pages 157–163, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[22] Robert A Peterson. A meta-analysis of variance accounted for and factor loadings in exploratory factor analysis. *Marketing letters*, 11(3):261–275, 2000.

[23] Keenan A Pituch and James P Stevens. *Applied multivariate statistics for the social sciences: Analyses with SAS and IBM's SPSS*. Routledge, 2015.

[24] John Ruscio and Brendan Roche. Determining the number of factors to retain in an exploratory factor analysis using comparison data of known factorial structure. *Psychological assessment*, 24(2):282, 2012.

[25] Susan Wiedenbeck, Vikki Fix, and Jean Scholtz. Characteristics of the mental representations of novice and expert programmers: an empirical study. *International Journal of Man-Machine Studies*, 39(5):793–812, 1993.

[26] Greg Wilson. *Teaching Tech Together: How to Make Your Lessons Work and Build a Teaching Community around Them*. CRC Press, 2019.

[27] Guojing Zhou, Jianxun Wang, Collin F Lynch, and Min Chi. Towards closing the loop: Bridging machine-induced pedagogical policies to learning theories. *International Educational Data Mining Society*, 2017.

LANA: Towards Personalized Deep Knowledge Tracing Through Distinguishable Interactive Sequences *

Yuhao Zhou
Sichuan University, China
sooptq@gmail.com

Xihua Li
Tencent Inc. China
lixihua9@126.com

Yunbo Cao
Tencent Inc. China
yunbocao@tencent.com

Xuemin Zhao
Tencent Inc. China
xueminzhao@tencent.com

Qing Ye
Sichuan University, China
fuyeking@gmail.com

Jiancheng Lv
Sichuan University, China
lvjiancheng@scu.edu.cn

ABSTRACT

In educational applications, *Knowledge Tracing* (KT) has been widely studied for decades as it is considered a fundamental task towards adaptive online learning. Among proposed KT methods, Deep Knowledge Tracing (DKT) and its variants are by far the most effective ones due to the high flexibility of the neural network. However, DKT often ignores the inherent differences between students (e.g. memory skills, reasoning skills, ...), averaging the performances of all students, leading to the lack of personalization, and therefore was considered insufficient for adaptive learning. To alleviate this problem, in this paper, we proposed **L**eveled **A**ttentive K**N**owledge Tr**A**cing (LANA), which firstly uses a novel student-related features extractor (SRFE) and pivot modules to distill and distinguish students' unique inherent properties from their respective interactive sequences. Moreover, inspired by Item Response Theory (IRT), the interpretable Rasch model was used to cluster students by their ability levels, and thereby utilizing leveled learning to assign different encoders to different groups of students. With pivot module reconstructed the decoder for individual students and leveled learning specialized encoders for groups, personalized DKT was achieved. Extensive experiments conducted on two real-world large-scale datasets demonstrated that our proposed LANA improves the AUC score by at least 1.00% (i.e. EdNet ↑ 1.46% and RAIEd2020 ↑ 1.00%), substantially surpassing the other State-Of-The-Art KT methods.

Keywords

Education, Personalized Learning, Adaptive Learning, Knowledge Tracing, Machine Learning, Deep Learning

*A full version of this paper is available at `https://github.com/Soptq/LANA-pytorch/raw/main/LANA_EDM2021.pdf`

Yuhao Zhou, Xihua Li, Yunbo Cao, Xuemin Zhao, Qing Ye and Jiancheng Lv "LANA: Towards Personalized Deep Knowledge Tracing Through Distinguishable Interactive Sequences". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 602-608. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

1. INTRODUCTION

Knowledge Tracing (KT) aims to accurately retrieve students' knowledge states at a certain time by his past sequential exercising interactions. To evaluate KT's performance, it is asked to predict the correctness of students' future exercises with the retrieved knowledge states as Equation 1 represented.

$$P(r^{s_i}_{t+1}|I^{s_i}_1, I^{s_i}_2, I^{s_i}_3, ..., I^{s_i}_t, \{I^{s_i}_{t+1} - r^{s_i}_{t+1}\}),\ I^{s_i}_t = (e^{s_i,q_j}_t, c^{q_j}, r^{s_i}_t) \tag{1}$$

Where e^{s_i,q_j}_t is referred as student $s_i \in \mathbb{N}^+$ answering question $q_j \in \mathbb{N}^+$ at discrete time step $t \in \mathbb{N}^+$, c^{q_j} represents the contextual information of question q_j (e.g. related concepts, part, etc.) [23, 14, 10, 4], and $r^{s_i}_t \in \{0, 1\}$ represents the correctness of student s_i's answer to q_j at time t. Additionally, the student's interaction sequence is defined as $S^{s_i}_{t_0,t_1} = \{I^{s_i}_t | t_0 < t < t_1\}$ and κ is defined as $\kappa^{s_i}_t = \{s_i, q_j, c^{q_j}, r^{s_i}_t\}$, referring to all features that participated in one interaction $I^{s_i}_t$ for latter explanation.

Traditionally, KT was regarded as a sequential behavior mining task [8, 17], and therefore various methods established models with the theory of bayesian probability (BKT [3]) and psycho-statistics (IRT [5]), providing excellent interpretability and good performance. Nevertheless, recently proposed Deep Knowledge Tracing (DKT) [16] and its variants [13, 4, 14, 1, 18] significantly outperform other KT methods in metrics using Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM [6]). However, DKT distinctly lacks personalization for students compared to BKT and IRT [15, 25], which are capable of separately training unique models for each student, while DKT only trains a unified model for all students due to massive training data and abundant computing resources required by deep learning. Hence, DKT weakly reflects the large inherent property (i.e. memory skills, reasoning skills, or even guessing skills) gaps between students.

ASSUMPTION 1. *For any interactive sequences satisfying* $\left|\left|S^{s_i}_{t_0,t_1}\right|\right| > \Theta >> 1$, $||\kappa|| > \Psi$ *and* $t_2 - t_1 > \mathcal{E}$, $S^{s_i}_{t_0,t_1}$ *can be distinguished from* $S^{s_j}_{t_0,t_1}$ *and* $S^{s_i}_{t_2,t_3}$ *respectively.*

Is it possible to bring personalization back to DKT? To answer this question, we observed that the proactive behavior

sequence (i.e. interactive sequences) of each individual is unique and changeable over time. Hence, we argue that the minimal personalized unit in KT is "a student at a certain time t_i" instead of just "a student", and **student's inherent properties at time t_i can be represented by his interactive sequences around time t_i (Assumption 1)**. In such a way, these student-related features could tremendously help personalize the KT process since they could be used to identify different students at different stages. Consequently, in our proposed **L**eveled **A**ttentive K**N**owledge Tr**A**cing (LANA), unique student-related features are distilled from students' interactive sequence by a Student-Related Features Extractor (SRFE). Moreover, inspired by BKT and IRT that assign completely different models to different students, LANA, as a DKT model, successfully achieves the same goal in a different manner. Detailedly, instead of separately training each student a model like BKT and IRT, **LANA learns to learn correlations between inputs and outputs on attention of the extracted student-related features**, and thus becomes transformable for different students at different stages. More specifically, the transformation was accomplished using pivot module and leveled learning, where the former one is a model component that seriously relies on the SRFE, and the latter one is a training mechanism that specializes encoders for groups with interpretable Rasch model defined ability levels. Formally, the LANA can be represented by:

$$\overbrace{r_t^{s_i} \sim (f(p_t^{s_i}))(h_t^{s_i})}^{Adaptive\ by\ Pivot\ Module} \quad , \quad \underbrace{p_t^{s_i} \sim k(h_t^{s_i}), \quad h_t^{s_i} \sim g(h_{<t}^{s_i}, S_{0,t}^{s_1})}_{Adaptive\ by\ Leveled\ Learning}, \tag{2}$$

where $h_t^{s_i}$ is referred as student s_i's knowledge state at time t respectively, $f(\cdot)$ (decoder), $g(\cdot)$ (encoder) and $k(\cdot)$ (SRFE) are three main modules that LANA seeks to learn.

2. METHODOLOGY

2.1 Base Modifications

There are mainly two base modifications in the LANA model (Figure 1) that were made to the basic transformer. Firstly, in the LANA model, the positional information (e.g. positional encoding, positional embedding) was directly fed into the attention module with a private linear projection, instead of being added to the input embedding and shared the same linear projection matrix with other features in the input layer. Although experiments in [22] suggested that blending input embedding with positional information is effective, recently some work [19] debated that when the model becomes deeper, it tends to "forget" the positional information fed into the first layer. Moreover, some other work [9] believed that adding positional information to the input embedding and offering them to the attention module, is essentially making them share the same linear projection matrix, which is not reasonable since the effects of the input embedding and the positional information are clearly distinctive. For exactly the same reason, in the LANA model, multiple input embeddings (i.e. question ID embedding, student ID embedding, etc.) are concatenated instead of added, leading to the second base modification. Specifically, assumes there are m input embeddings in total, each with a dimension of D^f. Then after concatenating, the input embedding would have a total dimension of D^{mf}. Hence, a $D^{mf} \to D^f$ linear projection layer was used to map the concatenated input embedding of dimension D^{mf} to dimension D^f.

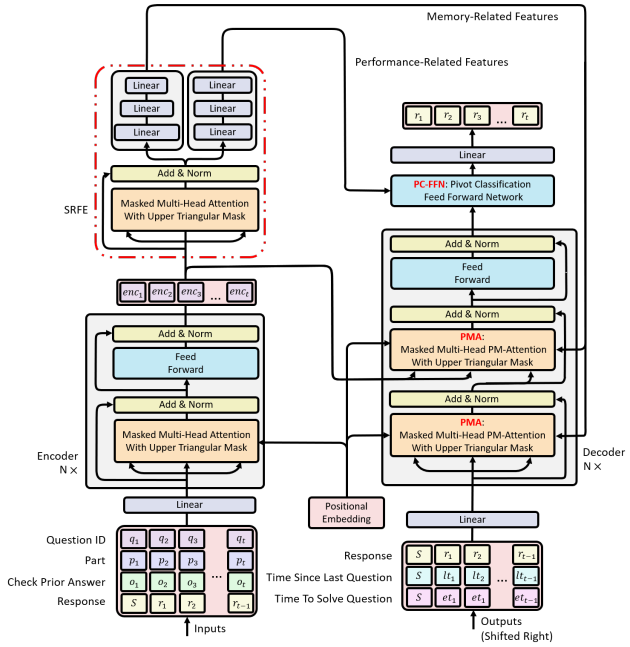


Figure 1: The overall model architecture of LANA. There are mainly three differences compared to vanilla transformer-based KT method [1, 18]: I. Modifications to the basic transformer model. II. Introduced SRFE and III. Introduced PMA Module and PC-FFN Module, which collectively referred to as pivot module.

2.2 Student-Related Features Extractor (SRFE)

Student-Related Features Extractor (SRFE) summaries students' inherent properties from their interactive sequences with Assumption 1 for the pivot module to personalize the parameters of the decoder. Specifically, SRFE contains an attention layer and several linear layers, where the attention layer was used to distill student-related features from the provided information by the encoder, and the linear layers were leveraged to refine and reshape these features. It is notable that in the LANA model there were primarily two SRFEs: memory-SRFE and performance-SRFE, where the former one was utilized to derive students' memory-related features for the PMA module (be introduced later) and the latter one was dedicated to distill students' performance-related features (i.e. Logical thinking skill, Reasoning skill, Integration skill, etc.) for PC-FFN module (introduced later either). The reshaping process was drawn in Figure 3 for better illustration, where bs, n_{heads}, seq and d_{piv} are referred to as the model's batch size, the number of attention heads [22], the length of the input sequence and the dimension of performance-related features. The intuition that memory-related features have a second dimension of n_{head} comes from the theory that each attention head only pays attention to one perspective of the features. Thus it is reasonable that each student has different memory skills for different attention heads (e.g. for different concepts).

2.3 Pivot Module

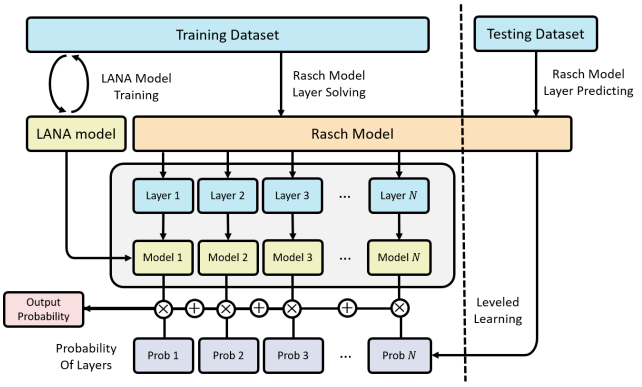


Figure 2: The workflow of leveled learning: interpretable Rasch model was leveraged to analyze students' overall ability levels, and then cluster students into multiple layers, where each layer would respectively fine-tune the LANA model by its own training data.

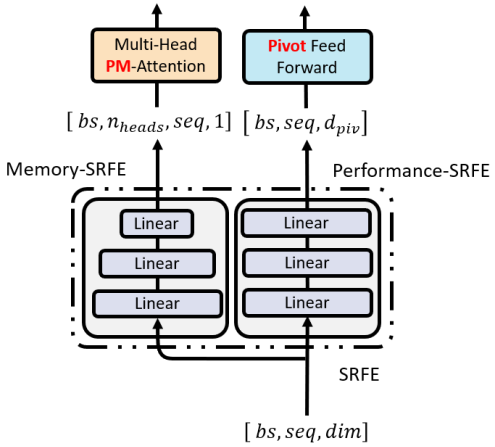


Figure 3: The data shape transformation of two SRFE: Memory-SRFE and Performance-SRFE.

Provided an ordinary input x, a student-related features p and a target output y, pivot module learns the process of learning how to project x to y based on p, instead of simply learning to project x to y (i.e. Pivot module learns to learn) as Equation 3 shown.

$$y = (f(p))(x), \tag{3}$$

where $f(\cdot)$ here is the function that pivot module learns to learn. That is, the projection matrix of x is adapted to p instead of being fixed. To accomplish this dynamic mapping, the weight and bias of x need to be a projection from p. Assumes $p \in \mathbb{R}^{D_p}$, $x \in \mathbb{R}^{D_x}$ and $y \in \mathbb{R}^{D_y}$, Equation 3 could be formally presented in Equation 4:

$$y = W^x x + b^x, \tag{4}$$

where $W^x \in \mathbb{R}^{D_y \times D_x}$ and $b^x \in \mathbb{R}^{D_y}$. Since W^x and b^x is derived from p, the detailed transformation could be revealed in Equation 5, which was also depicted in Figure 4 for better illustration.

$$W^x = W_1^p p + b_1^p, \quad b^x = W_2^p p + b_2^p, \tag{5}$$

where $W_1^p \in \mathbb{R}^{(D_y \times D_x) \times D_p}$, $b_1^p \in \mathbb{R}^{(D_y \times D_x)}$, $W_2^p \in \mathbb{R}^{D_y \times D_p}$ and $b_2^p \in \mathbb{R}^{D_y}$.

By simplification, Equation 3 can be defined as Equation 6, being named as $PivotLinear(x, p)$.

$$y = (Wp)x + b = PivotLinear(x, p), \tag{6}$$

where $W \in \mathbb{R}^{D_y \times D_x \times D_p}$ and $b \in \mathbb{R}^{D_y}$.

In the LANA model, there are primarily two modules that pertain to the pivot module: Pivot Memory Attention (PMA) Module and Pivot Classification Feed Forward Network (PC-FFN) Module. In many methods [4, 14], Vanilla Memory Attention (VMA) Module was employed to consider the "forgetting" behavior of students, which is pivotal in KT's context since students are very likely to have done similar exercises to the one he is going to do, and if the student could remember the answers to previous similar exercises, the probability of him correctly answering the future related exercises will be increased greatly. Inspired by the Ebbinghaus Forgetting Curve [12] and much previous work [14, 4], "forgetting" behavior of students are defined as exponentially decaying weights of corresponding interactions in the timeline. Detailedly, in the original attention module, the weight of item j on item k, i.e. $\alpha_{j,k}$, is determined by the `sigmoid` result of the similarity between item j and item k:

$$\alpha_{j,k} = \frac{sim(j,k)}{\sum_{k'} sim(j,k')}, \tag{7}$$

where $sim(\cdot)$ is a function to calculate the similarity between item i and item j by dot production. In order to take "forgetting" behavior into $\alpha_{j,k}$'s account (e.g. The further away from j, the lower the weight $\alpha_{j,k}$ would be), we replaced Equation 7 with Equation 8:

$$\alpha_{j,k,m} = \frac{e^{-(\theta+m)\cdot dis(j,k)} \cdot sim(j,k)}{\sum_{k'} sim(j,k')}, \tag{8}$$

where m is the student's memory-related features extracted in memory-SRFE, θ is a private learnable constant that describes all students' average memory skill in the PMA module, and $dis(\cdot)$ calculates the time distance between item j and item k (e.g. item j is done $dis(j,k)$ minutes after item k is done). The reason for representing the memory skill with two learnable parameters is to reduce the difficulty for model converging since m has a much longer back-propagation path compared to θ. When θ is introduced to fit

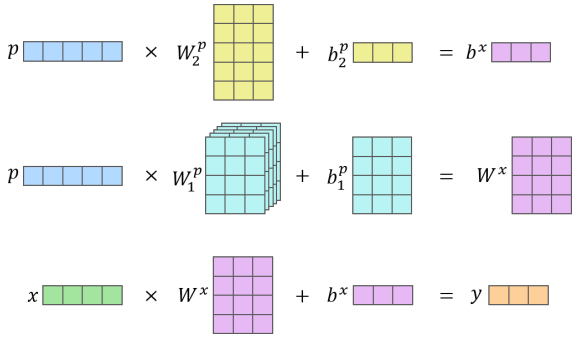


Figure 4: An illustration of the data transformation in the pivot module.

the average memory skill of all students, the distribution of m becomes a Gaussian distribution, which makes the model much easier to learn.

On the other hand, PC-FFN was utilized to make the final prediction in reference to the performance-related features, which essentially is a PivotLinear module with a dropout and activation. The idea of this module comes from many investigations that the early layers in a deep neural network are often used as a feature extractor while the latter layers are often used as a decision-maker to decide which feature is useful to the output of the model. As a result, these investigations point out that many models are actually having similar early layers, and it is the latter layers that make these models distinctive in usage. Consequently, PC-FFN in the LANA model was utilized as a personalized decision-maker to adaptively make the final prediction based on students' distinctive inherent properties:

$$PC{-}FFN(x,\, p) = x + PivotLinear(PivotLinear(x,\, p),\, p), \tag{9}$$

where p is the students' performance-related features extracted in the performance-SRFE.

2.4 Leveled Learning

While the pivot module enables the decoder to be transformable for different students, the encoder and the SRFE of the LANA model that provides necessary information for the pivot module remains the same for all students. This is not problematic if the length of the input sequence is large enough since Assumption 1 assures long sequences are always distinguishable, unless they both belong to the same student at the same time period. However, DKT, especially transformer-based DKT, can only be inputted with the latest n (commonly $n = 100$) interactions at once due to the limited memory size and high computational complexity. Consequently, it is possible for the encoder and SRFE to output similar results for two different students, resulting in a failure for the decoder to adapt. To alleviate this problem, it is natural to think of assigning different students with different encoders and SRFEs that are highly specialized (sensitive) to their assigned students' patterns. However, in practice, it is not feasible to train a unique encoder for each individual student considering both the limited training time and the limited training data. As a result, a novel leveled learning (Figure 2) method was proposed to address this problem, which was initially inspired by the fine-tuning mechanism in transfer learning [20], where we consider each student a unique task, and we want to transfer a model that fits well on all students to one student s_i efficiently.

Leveled learning holds the view that the earlier layers of a model are similar for similar tasks. Thus, to save training time and enlarge the training set, instead of training each student a unique encoder and SRFE by his private training data, students with similar ability levels are considered to be grouped together, sharing their private training data and having the same encoder and SRFE. Therefore, LANA firstly utilizes an interpretable Rasch model to analyze the ability level a^{s_i} for each student s_i, then groups students into different independent layers l_i. Assuming the ability distribution of all students and students at the level l_i are Gaussian distribution $N(\mu_a, \sigma_a^2)$ and $N(\mu_i, \sigma_i^2)$ respectively, we have the Equation 10:

$$\mu_a = \frac{\sum_i \mu_i}{L}, \quad \sigma_a^2 = \sum_i \sigma_i^2. \tag{10}$$

In LANA, for simplicity, we consider all layers share the same variance σ^2 [1], and the difference of mean μ_i between consecutive layers is a constant τ. Hence, μ_i and σ_i^2 are given by:

$$\mu_i = \mu_a - \frac{L-1}{2} \times \tau + i \times \tau, \quad \sigma_i^2 = \frac{\sigma_a^2}{L}. \tag{11}$$

where $L = ||l_i||$ is the number of layers. With both μ_i and σ_i^2 retrieved for every layer l_i, given a student's ability constant a^{s_i}, we can now calculate the probability of s_i been grouped into different layers by Equation 12:

$$p_i^{s_i} = \frac{\phi_i(a^{s_i})}{\sum_{i'} \phi_{i'}(a^{s_{i'}})}, \quad \phi_i(a^{s_i}) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(a^{s_i} - \mu_i)^2}{2\sigma_i^2}} \tag{12}$$

where $p_i^{s_i}$ is referred as the probability of student s_i being grouped into layer l_i. As it can be seen from Equation 12, students that have high ability levels are not necessarily grouped into layers with high expected ability levels μ_i. Contrarily, these high ability students only have a higher probability of been grouped into high ability layers in comparison with those low ability students, which obeys rules in reality (e.g. high ability students may also come from normal schools).

Then, the LANA model that has been pre-trained on all students was duplicated L times, each cloned model m_i would be assigned to a layer l_i to be dedicatedly fine-tuned with l_i's private training data by weighted back-propagation:

$$loss_i = p_i \times loss(predict_i, target), \tag{13}$$

where $predict_i$ is the prediction of the model m_i.

While the training phase of leveled learning seems promising, the inference phase of it suffers problems. The first problem is how to make the prediction using multiple specialized models. In LANA, the prediction was made by $top-k$ models fusion. Detailedly, when student s_i's future responses are needed to be predicted, LANA firstly computes p_i, then feed s_i's interactive sequence to all models m_i that satisfies $p_i \in top-k(p)$, where k needs to be manually set up to control the predicting time. Then, the outputs of these models would be multiplied by $sigmoid(p_i)$ to form the final prediction. The workflow of leveled learning's inference step could be described in Equation 14:

$$r_i = \sum_{i'} (m_i(x) \times \sum_{h \in i'} \frac{p_h}{\sum_{h'} p_{h'}}), \quad i' \in \{\, i \mid p_i \in top-k(p) \,\}, \tag{14}$$

where r_i is the leveled learning's final prediction and x is the input of the model. This workflow seems similar to the ensemble where multiple models are unitized to generate the final answer. Nonetheless, weights of models in LANA are probabilities that come from an interpretable Rasch model

[1] In practice, if the number of layers is small, their variances then need to be manually measured and tuned based on the targets. If the number of layers is large, then multiple layers can be regarded as one layer and therefore sharing the same variance for all layers should be fine.

so that it is clear which model is dominant to x. Moreover, unlike in ensemble, where the role of each model is ambiguous, in LANA, every model has its explainable effect (e.g. l_L is committed to high ability students, and therefore a student with large p_L indicates he must be similar to those high ability students in l_L), suggesting that leveled learning significantly outperforms ensemble in interpretability. Detailed comparison was shown in Table 1. On the other hand, the

Table 1:
Comparison Between Leveled Learning And Ensemble

	Leveled Learning	Ensemble
Sub-set Select	Psycho-statistics	Random
Interpretability	Good	Bad
Predicting Time	Controllable (top-k)	Uncontrollable

second problem of the leveled learning is how to compute p_i for students that LANA has never met in training, namely the "cold start" problem [24]. In vanilla KT context, we can only initiate newly arrived students' ability levels to the average ability level of all students. However, in practice, we can estimate their ability levels more accurately by asking them to do a couple of sample exercises or using ranking at school.

3. EXPERIMENTS

3.1 Experimental Setup

In order to evaluate the effectiveness of the proposed LANA[2], we applied it to two real-world large-scale datasets in comparison with many other State-Of-The-Art (SOTA) KT methods. Specifically, EdNet [2] and RAIEd2020 [7] are employed in our experiments, where EdNet is currently the largest publicly available benchmark dataset in education domain, consisting of over 90,000,000 interactions and nearly 800,000 students. On the other hand, RAIEd2020 is a recently published real-world dataset that has approximately the same size as EdNet with nearly 100,000,000 interactions and 400,000 students. Particularly, the average number of exercising interactions per student in RAIEd2020 is double to EdNet's. Moreover, 6 KT methods that had previously achieved SOTA performance have participated in the comparison: DKT [16], DKVMN [26], SAKT [13], SAINT [1], SAINT+ [18], AKT [4]. In terms of the basic experimental environment, all experiments were conducted with Pytorch[3] 1.6 on a Linux server that is equipped with an Nvidia V100 GPU. For hyper-parameters setup, the learning rate was set to $5e-4$ with *AdamW* [11] optimizer, the length of the input sequence was set to 100, the batch size was set to 256, and other detailed configurations were listed in our source code. The input features κ in EdNet contains *Question ID*, *Question part*, *Students' responses*, *Time interval between two consecutive interactions* and *Elapsed time of an interaction*, whereas in RAIEd2020, a new feature is additionally added to κ, which indicates *Whether or not the student check the correct answer to the previous question.* Finally, The Area Under the receiver operating characteristic Curve (AUC) was leveraged in our experiments as the performance metric, which has been widely used in many other KT-related proposals.

[2] https://github.com/Soptq/LANA-pytorch
[3] https://pytorch.org/

Table 2:
The AUC Comparison Of Different Methods Tested On EdNet And RAIEd2020 datasets

Dataset	Model	AUC
EdNet	DKT	0.7638^r
EdNet	DKVMN	0.7668^r
EdNet	SAKT	0.7663^r
EdNet	SAINT	0.7816
EdNet	SAINT+	*0.7913*
EdNet	SAINT+ & *BM*	0.7935
EdNet	LANA	**0.8059**
RAIEd2020	SAKT	0.7832
RAIEd2020	AKT	0.7901
RAIEd2020	SAINT+	*0.7956*
RAIEd2020	SAINT+ & *BM*	0.7991
RAIEd2020	LANA	**0.8056**

Table 3:
Investigation Of The Effectiveness Of Different Improvements In LANA

Dataset	BM	Pivot Module		LL	AUC	Boost
		PMA	PC-FFN			
EdNet					0.7913	-
EdNet	✓				0.7935	↑ 0.0022
EdNet		✓			0.7997	↑ 0.0084
EdNet			✓		0.7923	↑ 0.0010
EdNet				✓	0.7933	↑ 0.0020
EdNet	✓	✓			0.8029	↑ 0.0116
EdNet		✓	✓		0.8015	↑ 0.0102
EdNet	✓	✓	✓		0.8038	↑ 0.0125
EdNet	✓	✓		✓	0.8050	↑ 0.0137
EdNet	✓	✓	✓	✓	**0.8059**	↑ **0.0146**
RAIEd2020					0.7956	-
RAIEd2020	✓				0.7991	↑ 0.0035
RAIEd2020		✓			0.8020	↑ 0.0064
RAIEd2020			✓		0.7965	↑ 0.0009
RAIEd2020				✓	0.7977	↑ 0.0021
RAIEd2020	✓	✓			0.8031	↑ 0.0075
RAIEd2020		✓	✓		0.8027	↑ 0.0071
RAIEd2020	✓	✓	✓		0.8035	↑ 0.0079
RAIEd2020	✓	✓		✓	0.8051	↑ 0.0095
RAIEd2020	✓	✓	✓	✓	**0.8056**	↑ **0.0100**

For the ease of explanation, hereinafter Base Modification (Section 2.1), Pivot Module (Section 2.3) and Leveled Learning (Section 2.4) would be abbreviated as *BM*, *PM* and *LL* respectively.

3.2 Results And Analysis

The overall experimental results of different KT methods on different datasets were illustrated in Table 2. Because we had successfully reproduced the performance of SAINT and SAINT+ that was previously reported in SAINT+'s paper [18] (with considerable precision), AUCs of other models are therefore directly cited from the paper (labeled with subscript r).

From the comparison table, it can be seen that in both Ed-

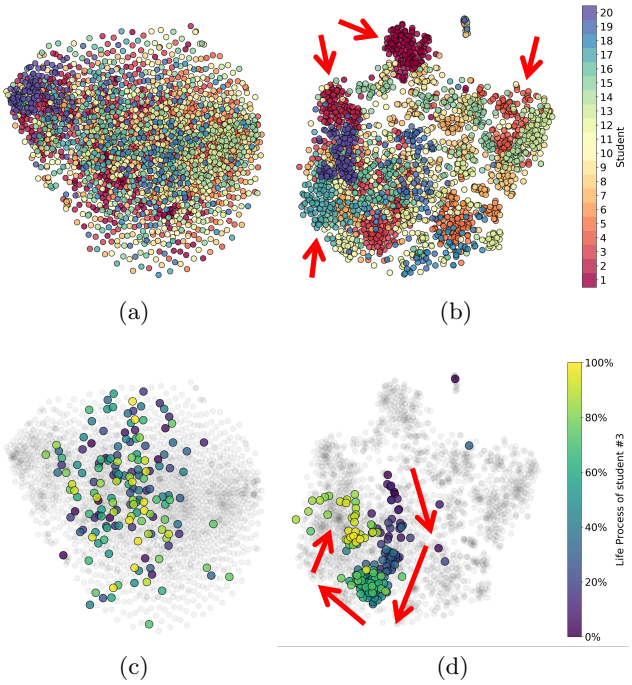


Figure 5: The visualization of intermediate features in SAINT+ (a) and in LANA (b). Compared to (a), students in (b) (different colors) are notably clustered (marked arrows). The learning process of student #3 overtime in SAINT+ (c) and in LANA (d). compared to (c), a clear learning path appeared in (d).

Net and RAIEd2020 datasets, LANA (marked bold) outperforms the previous SOTA method (marked italic) by 1.46% and 1.00% respectively, readily verifying the effectiveness of our proposed improvements. Moreover, LANA also surpasses SAINT+ & *BM* by 1.24% and 0.65% respectively, suggesting adaptability contributes most to LANA's AUC increment. Considering experimented datasets are by far the two largest knowledge tracing datasets in the world, these results undoubtedly provide strong evidence of the validity of the proposed LANA method.

3.3 Ablation Studies

In this section, we investigated the effectiveness of each of our proposed improvements: *BM* that customizes the basic transformer architecture, *PM* that enables the decoder to be adaptive to the students' personal characteristics, and *LL* that interpretably specializes encoders and SRFEs for better predicting performance. The results of the ablation study were shown in Table 3.

The table shows in EdNet, applying *BM* alone was already capable of improving the predicting AUC by approximately 0.2% averagely, verifying the importance of both the action of positional embedding and the personalized linear projection for each input feature in KT's context. Meanwhile, applying *LL* solely can benefit the model performance as well, by generally 0.2% compared to 0.1% with the vanilla ensemble. Considering without *PM*, *LL* would just perform fitting on students with different ability levels, the performance gain from sole *LL* could be interpreted as reductions in students' inherent properties gaps. Moreover, *BM* + *PM* drastically boosts the model performance by nearly 1.25%, suggesting *PM* makes proper use of extracted student-related features from SRFE to adaptively reparameterize the model's decoder for different students at different stages, and therefore contributes most to the final performance gain. Finally, by combining all improvements together, *BM* + *PM* + *LL* (i.e. LANA) achieves a final AUC of 0.8059, substantially outperforms previous SOTA by at least 1.46%.

3.4 Features Visualization

For vividly illustrating the validity of student-related features distilling in LANA, 20 students' intermediate features from PC-FFN module was sampled to generate Figure 5 by t-SNE[21]. In figure 5 (a) and (b), each sample represents intermediate features of different students with different colors in SAINT+ and LANA respectively. It can be seen that in SAINT+, samples are almost randomly distributed, indicating the correlation between samples of the same student is not more significant compared to samples of the others due to the ignorance of students' personalities. On the other hand, in LANA, clusters (marked arrows) of samples have notably appeared in comparison to (a). Thus, we concluded that LANA is capable of successfully extracting student-related features from their interactive sequences, summarizing the similarities and differences, which eventually results in more distinguishable features for the final classifier.

Furthermore, we individually visualized student #3's (randomly picked) samples along the time axis to investigate the transitioning pattern of features in Figure 5 (c)(SAINT) and (d)(LANA). In (c), there is no clear pattern in the change of features over time, while in (d), a clear transitioning path could be noticed. Since many other students are sharing the same pattern in LANA, we argue that it represents the trajectory of the student's ability changes with more and more exercising. Namely, it is the learning path of the student. Consequently, we contended that it is potentially helpful for other applications, such as learning stages transfer and learning path recommendation.

4. CONCLUSION

In this paper, we proposed a novel **L**eveled **A**ttentive K**N**owledge Tr**A**cing (LANA) method that was committed to bringing adaptability back to DKT. Instead of directly learning the model parameters of different students, LANA distills students' inherent properties from their respective interactive sequences by a novel SRFE, and learns the function to reparameterize the model with these extracted student-related features. Consequently, innovative pivot module was proposed to produce an adaptive decoder. Besides, a novel leveled learning training mechanism was introduced to cluster students by interpretable Rasch model defined ability level, which not only specializes the encoder and therefore enhances the significance of students' latent features, but also saves much training time. Extensive experiments on the two largest public benchmark datasets in the education domain strongly evaluate the feasibility and effectiveness of the proposed LANA, features visualization also suggests extra impacts of LANA, be it learning stages transfer or learning path recommendation.

5. REFERENCES

[1] Y. Choi, Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and J. Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pages 341–344, 2020.

[2] Y. Choi, Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, C. Bae, B. Kim, and J. Heo. Ednet: A large-scale hierarchical dataset in education. In *International Conference on Artificial Intelligence in Education*, pages 69–73. Springer, 2020.

[3] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[4] A. Ghosh, N. Heffernan, and A. S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2330–2339, 2020.

[5] J. González-Brenes, Y. Huang, and P. Brusilovsky. General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In *The 7th international conference on educational data mining*, pages 84–91. University of Pittsburgh, 2014.

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] R. Inc. Riiid aied challenge 2020. https://www.kaggle.com/c/riiid-test-answer-prediction/data, 2020. [Online; accessed 6-Oct-2020].

[8] A. Jalal and M. Mahmood. Students' behavior mining in e-learning environment using cognitive processes with information technologies. *Education and Information Technologies*, 24(5):2797–2821, 2019.

[9] G. Ke, D. He, and T.-Y. Liu. Rethinking the positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020.

[10] Q. Liu, Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, and G. Hu. Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, 33(1):100–115, 2019.

[11] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[12] J. M. Murre and J. Dros. Replication and analysis of ebbinghaus' forgetting curve. *PloS one*, 10(7):e0120644, 2015.

[13] S. Pandey and G. Karypis. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*, 2019.

[14] S. Pandey and J. Srivastava. Rkt: Relation-aware self-attention for knowledge tracing. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1205–1214, 2020.

[15] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 255–266. Springer, 2010.

[16] C. Piech, J. Spencer, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. *arXiv preprint arXiv:1506.05908*, 2015.

[17] S. Shang, L. Chen, C. S. Jensen, J.-R. Wen, and P. Kalnis. Searching trajectories by regions of interest. *IEEE Transactions on Knowledge and Data Engineering*, 29(7):1549–1562, 2017.

[18] D. Shin, Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi. Saint+: Integrating temporal features for ednet correctness prediction. *arXiv preprint arXiv:2010.12042*, 2020.

[19] V. L. Shiv and C. Quirk. Novel positional encodings to enable tree-based transformers. In *NeurIPS*, pages 12058–12068, 2019.

[20] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[21] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[23] Z. Wang, X. Feng, J. Tang, G. Y. Huang, and Z. Liu. Deep knowledge tracing with side information. In *International conference on artificial intelligence in education*, pages 303–308. Springer, 2019.

[24] K. H. Wilson, X. Xiong, M. Khajah, R. V. Lindsey, S. Zhao, Y. Karklin, E. G. Van Inwegen, B. Han, C. Ekanadham, J. E. Beck, et al. Estimating student proficiency: Deep learning is not the panacea. In *In Neural Information Processing Systems, Workshop on Machine Learning for Education*, page 3, 2016.

[25] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*, pages 171–180. Springer, 2013.

[26] J. Zhang, X. Shi, I. King, and D.-Y. Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*, pages 765–774, 2017.

Recommendation System for Engineering Programs Candidates

Bruno Mota da Silva
Instituto Superior Técnico
brunomotadasilva@tecnico.ulisboa.pt

Cláudia Antunes
Instituto Superior Técnico
claudia.antunes@tecnico.ulisboa.pt

ABSTRACT

Automatic discovery of information in educational data has been broadening its horizons, opening new opportunities to its application. An open wide area to explore is the recommendation of undergraduate programs to high school students. However, traditional recommendation systems, based on collaborative filtering, require the existence of both a large number of items and users, which in this context are too small to guarantee reasonable levels of performance.

In this paper, we propose a hybrid approach, combining collaborative filtering and a content-based architecture, while exploring the hierarchical information about programs organization. This information is extracted from courses programs, through natural language processing, and since programs share some courses, we are able to present recommendations, not just based on the performance of students, but also on their interests and results in each of the courses that compose each program.

Keywords

Recommendation systems, higher education programs, educational data mining

1. INTRODUCTION

Nowadays, it is common to have teenagers applying to a higher education program after finishing their high school. Every year, new programs appear and thousands of candidates must choose which one is the best for them.

This type of problem is very well-known in Educational Data Mining and in Recommendation Systems community [3, 11]. This past decade, many studies were made on creating engines that help students in choosing the courses that are suited for them, using different approaches, like content-based or collaborative filtering recommendation systems. The last type is the most used due to the large amount of data community can give.

Bruno Mota da Silva and Claudia Antunes "Recommendation System for Engineering Programs Candidates". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 609-613. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Despite courses recommendation being a more studied problem, we want to apply these systems to programs recommendation that is not very researched yet. This brings an important challenge, since courses recommenders have already the target user inside the system rating previous courses among the others students, and in our problem candidates did not rate anything to be compared to other users in first hand.

Considering all of these aspects, our work aims for creating a recommendation system that will receive candidates personal data and high-school academic records, with the proper consent given by them considering general data protection regulations (GDPR), and will output the programs that most fit to their profile, comparing to the current student community. The system will consider the personal characteristics of the students as a matching measure and the programs' courses, objectives and description to find keywords that define the corresponding programs. These keywords will allow to compute ratings for every program considering the academic marks of the students on their own program.

This paper is divided in four more sections. Literature review covers the basic aspects of recommendation systems, with special focus on their use for educational purposes. After this, we present the architecture of our system that can be applied at a common university structure. After system architecture, current results are shown, followed by the reached conclusions at this time.

2. LITERATURE REVIEW

Recommendation Systems (RS) are software tools and techniques that provide suggestions for items to be of use to a user [10]. A RS can be exploited for different purposes, such as, to increase the number of items sold, to better understand what the user wants or, in another point of view, to recommend a specific item to that user.

There are two main types of recommendation engines, Content-based and Collaborative Filtering. The first one is focused on item similarities, and the second one use past behaviors of users to recommend items to the active user [1].

There is also a third type of recommendation systems, knowledge-based approaches where recommendations are given based on explicit specification of the kind of content the user wants. These systems are very similar to content-based ones, but with domain knowledge input. Finally, a hybrid recommen-

dation system is constructed if there is a combination of two or more RS philosophies in order to improve the global performance.

Over the years, a large amount of educational data is being generated and there are being applied more collaborative filtering approaches than content-based methods in this area.

Morsomme and Alferez proposed a collaborative recommendation system that outputs courses to the target users, by exploiting courses that other similar students had taken, through k-means clustering and K-nearest neighbors techniques [2].

A recommendation system for course selection was developed in Liberal Arts bachelor of the University College Maastricht [6], using two types of data, students and courses. Student data consisted of anonymized students' course enrollments, and course data consisted of catalogues with descriptions of all courses, which allowed to find the topics of each one, using the Latent Dirichlet Allocation statistical model. Recurring to regression models of student data, the authors could predict his grade for each course. In the end, the system outputs 20 courses whose content best matches the user's academic interest in terms of Kullback-Leibler distance.

This content-based approach was applied as well in Dublin [8], where the authors used an information retrieval algorithm to compute course-course similarities, based on the text description and learning outcomes of each one.

In Faculty of Engineering of the University of Porto, it was created an engine to help students choosing an adequate higher education program to access a specific job in the future [5]. Therefore, it was implemented a recommendation system that uses the data from alumni and job offers and outputs a ranking of programs that could lead to the candidates' desired careers. The collaborative filtering approach can match the skills needed for that job and the skills given to the students of a specific degree.

Fábio Carballo made an engine that predicts students masters courses marks, using collaborative filtering methods, singular value decomposition (SVD) and as-soon-as-possible (ASAP) classifiers. With his work, he could recommend the more suitable program for students skills [4].

The topic around course and programs recommendations gained even more attention recently, with several published studies in the last years, following a variety of approaches [13, 14, 7, 9, 12].

3. RECOMMENDATION SYSTEM

The proposed system shall enlighten candidates about the degrees that are more compatible with their interests and that were successfully concluded by similar students, using a hybrid approach.

Our system must recommend higher education programs to a specific high-school student who wants to enroll at university. Usually, the candidate searches information about each program at universities web pages, such as courses or professional careers, or talks with students who are already enrolled at the programs he or she likes. The process of choosing a degree is very important to a high school student and it must be done analysing all the information available. Therefore, the main use case of our system focuses on candidates point of view.

As we can see on Figure 1, when the candidate uses our system, he or she must be able to give personal data that will be considered during the recommendation process. After that, the system must output a ranking of the programs that are most suitable to the candidate. Candidate's personal data can be academic interests, high school grades, personal data, such age or gender, among others. Since we are collecting data, it must be made according to the GDPR, applying anonymization techniques when necessary.

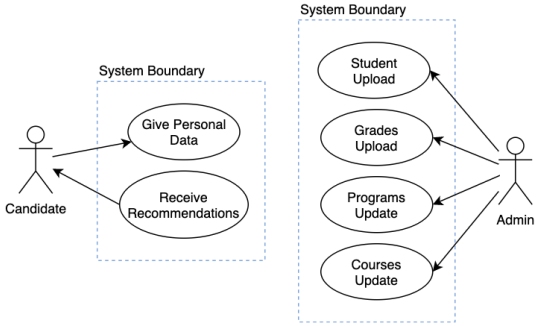


Figure 1: Use cases of the system.

Looking at the system from Admin point of view, there are several tasks he or she must be able to do, as we can see on Figure 1. System Administrator is the one responsible for system updates: upload new students data every year, upload students grades at the end of each semester, and update programs and courses when necessary. All the essential data to relate the candidate to current students and to make proper recommendations must be inputted before the system launching.

Finally, analysts staff can use this system when useful, to get a summary of student community and a characterization of new students.

3.1 Architecture

The overview of our system architecture can be seen on Figure 2, where we can distinguish two main modules: Students Profiler and Programs Recommender.

Candidates start using our system by inputting their personal data that will be used to find their profile. Current students data allow us to compute candidate profiles that will feed the second module. Programs Recommender uses the previous output to estimate a program success measure considering estimated grades, returning in the end a ranking of the most suitable programs to the candidates.

Figure 2: Proposed architecture.

3.1.1 Students Profiler

There is a major difference between our recommendation system and the common ones, where the target user is inside the system among the others. Here, the target user candidate is not in the system, since he or she is not enrolled at a higher education degree yet, and therefore can not rate programs or take courses. Hence, it must be developed a strategy where we can compare users.

Students Profiler computes the candidate profile as if he or she was inside the system, by comparing him with current students using shares personal variables. Therefore, the first step was to collect these data and to build a students profiling model, where we performed a feature engineering study.

A simple choice to implement Students Profiler is to apply the K Nearest Neighbors (KNN) method, after choosing the best similarity measure and number of neighbors, K. To compute the similarity, we used five different measures which results were studied. We also tuned the KNN process as well by trying to find the optimal value for K, that was the one having the minimum error rate.

In the end, Students Profiler returned candidate profile that will be fed to the next module.

3.1.2 Programs Recommender

Programs Recommender module is a more complex one which aims at finding the ranking of the best programs to the candidates, considering their profile **and interests**.

In order to reach its goal, this module has to create two models. The first one, called Grades Model, for estimating the candidate performance in each possible academic units, and the second one, the Ranking Model, for mapping students to programs.

As usual, the Grades Model is constructed by following a collaborative filtering approach, meaning that it uses a singular value decomposition (SVD) matrix factorization. This factorization performs a feature extraction step, reducing the number of elements to the minimum required for estimating students grades. When in the presence of the candidate profile, the Grades Model is applied to estimate the candidate grades. Using the candidate profile, instead of its original data, is the first difference in our approach, but there is more, achieved through the use of a content-based approach.

RS usually deal with a very large number of items, but the number of programs available in any university is just a few, when compared. Additionally, each student is enrolled on just one program, which means that our grades matrix would be very sparse, not contributing for a good recommendation. A third aspect is that programs share some courses (for example all engineering students study Physics and Maths, while all art students study Drawing and Geometry). But we can go a step further, and understand that courses cover some topics present in different areas. For example, several engineering courses study systems, their architecture and their dynamics.

The third proposal is the possibility of dealing with the academic units at different levels of granularity: we can aggregate everything to recommend programs, or we can simply identify a ranking of topics that are recommend for the candidate. This ability is very important to reach a new level of explainability, so needed in the field.

4. PRELIMINARY RESULTS

A recommendation system validation is a hard task to take. In contexts, like education, where these systems can not be made available before being proved 'correct', this task is even harder.

In our case, we made use of students data collected at the time of their enrollment in the university, to mimetize candidates surveys. Then, we used students data from 2014 to 2018 for training and data from 2019 for evaluation purposes. Moreover, every model of the system has to be validated independently, in order to better estimate each component performance, and only after tuning each of them evaluate its global quality.

We started by evaluating the Students Module, which has the use of KNN to estimate candidate profile on its basis. As data sources for this phase, we had personal data from 7918 students and grades from 7302 students, that resulted in a dataset of 7300 instances by intersecting the first ones. This dataset is composed by 101 variables, where enrolled program is the only categorical one, all of the others are numeric. Note that, we had no missing values on the dataset.

In this module, we wanted to find the K students that are most similar to the candidate. Therefore, we made a study to find the best pair (K, similarity measure) mimetizing a KNN performance study, but without focusing on the classification task. First, we needed to define which condition must students achieve to have success on their program, based on their Grade Point Average (GPA), from a 0-20 scale. Hence, a histogram was made and it is shown at Figure 3.

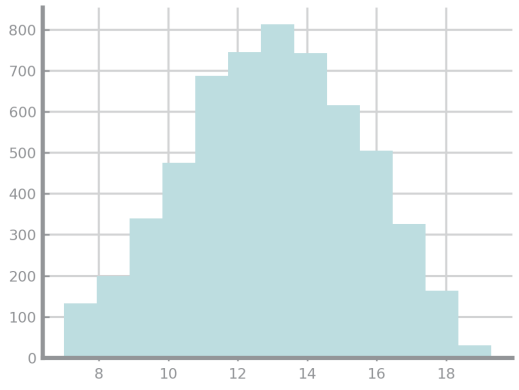


Figure 3: Number of students by each GPA class.

Since the average of students GPA is 12.99, we labeled as having success students which GPA was equal to 13 or more, and not having success otherwise. This way we guaranteed a balanced dataset. After the labelling, we computed ten trials

of data train-test split for five similarity measures (chebyshev, correlation, cosine, euclidean, and manhattan) and for K between 5 and 155 in multiples of 5. For each pair (K, similarity measure), we computed the average of KNN model accuracies, since 70% train and 30% test datasets are random in each trial. The results are shown in Figure 4, and zoomed in Figure 5.

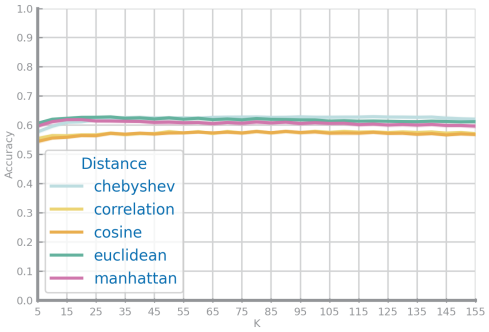


Figure 4: K and similarity measure study.

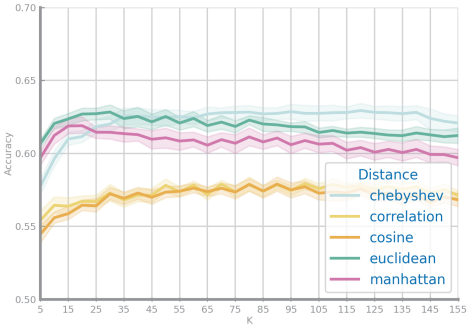


Figure 5: Zoom of K and similarity measure study.

Students Profiler module has five conditions that will be tested in the global system: (120, chebyshev); (100, correlation); (90, cosine); (30, euclidean) and (20, manhattan).

We implemented as well a simple recommendation system where we used the candidate profile, composed by the average grades of all neighbors for all courses taken by them, to predict the candidate grades for all available courses. In this component, four conditions were used for testing the system behaviour for all similarity measures: A) using SVD as matrix factorization technique with the K values mentioned above and considering all the variables from students data; B) same as A), but using K equals to 5; C) using SVD with the best K values predicted using a reduced students dataset with only academic records; and D) same as C) but using the Slope One prediction method.

After that, we used 1509 candidates to test the system, where we computed the GPA that each of them would have in each one of the available programs using their predicted course grades and ranked them by GPAs. Then, we computed the mean absolute error for those which first recommended program coincides with their current program in terms of GPA, and results are showed in Table 1.

Table 1: Mean Absolute Errors for each prediction method and for each similarity measure

Similarity Measure	A	B	C	D
chebyshev	2.065	2.378	2.139	2.289
correlation	2.149	2.580	2.359	2.325
cosine	2.153	2.583	2.430	2.451
euclidean	2.538	2.497	2.153	2.289
manhattan	2.313	2.488	2.376	2.451

The next steps will consist of improving the way we recommend the programs and its ranking model, considering different ensembles, namely random forests and gradient boosting. At this time, we are predicting GPA with almost 90% accuracy.

5. CONCLUSIONS

The current educational context, even more after the beginning of the pandemic situation, demands new educational systems. Systems able to address the difficulties inherent to distance learning contexts, where students are far from educators, and plenty of times try to follow their path without any guidance. Most of the times, online education tools deal with students in a 'one-fit-all' approach, that ignore each students preferences.

In this paper, we propose a new architecture for a recommendation system, designed for suggesting programs to university candidates. Our system benefits from an hybrid architecture, that combines collaborative filtering with a content-based philosophy, exploring the full documentation of programs and courses available. Additionally, we explored the notion of feature stores to easily update the data repositories to support our system.

The proposed architecture is adaptable to smaller contexts, for example for suggesting learning resources at any abstraction levels, such as exercises.

6. ACKNOWLEDGMENTS

This work was supported by national funds by Fundação para a Ciência e Tecnologia (FCT) through project GameCourse (PTDC/CCI-CIF/30754/2017).

7. REFERENCES

[1] C. C. Aggarwal. *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[2] A. Al-Badarneh and J. Alsakran. An automated recommender system for course selection. *International Journal of Advanced Computer Science and Applications*, 7, 03 2016.

[3] R. S. Baker and K. Yacef. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining*, 1(1):3–17, 2009.

[4] F. O. G. Carballo. Masters' courses recommendation: Exploring collaborative filtering and singular value

decomposition with student profiling. 2014.
[5] A. I. N. A. de Sousa. Market-based higher education course recommendation. 2016.
[6] R. Morsomme and S. V. Alferez. Content-based course recommender system for liberal arts education. 2019.
[7] B. MS, Y. Taniguchi, and S. Konomi. Course recommendation for university environments. 07 2020.
[8] M. P. O'Mahony and B. Smyth. A recommender system for on-line course enrolment: An initial study. page 133–136, 2007.
[9] A. Polyzou, A. N. Nikolakopoulos, and G. Karypis. Scholars walk: A markov chain framework for course recommendation. 05 2019.
[10] F. Ricci, L. Rokach, and B. Shapira. Recommender systems handbook. *Recommender Systems Handbook*, 1-35:1–35, 10 2010.
[11] A. Rivera, M. Tapia-Leon, and S. Luján-Mora. Recommendation systems in education: A systematic mapping study. pages 937–947, 01 2018.
[12] F. Scherzinger, A. Singla, V. Wolf, and M. Backenköhler. Data-driven approach towards a personalized curriculum. 07 2018.
[13] R. Yu, Q. Li, C. Fischer, S. Doroudi, and D. Xu. Towards accurate and fair prediction of college success: Evaluating different sources of student data. 07 2020.
[14] Y. Zhao, Q. Xu, M. Chen, and G. M. Weiss. Predicting student performance in a master of data science program using admissions data. 07 2020.

Deep learning for sentence clustering in essay grading support

Li-Hsin Chang, Iiro Rastas, Sampo Pyysalo, and Filip Ginter
TurkuNLP Group
Department of Computing
University of Turku
{lhchan, iitara, sampyy, figint}@utu.fi

ABSTRACT

Essays test student knowledge on a deeper level than short-answer and multiple-choice questions but are more laborious to evaluate. Automatic clustering of essays, or their fragments, prior to evaluation may reduce the manual effort required. Such clustering presents numerous challenges due to the variability and ambiguity of natural language. In this paper, we introduce two datasets of undergraduate student essays in Finnish, manually annotated for salient arguments on the sentence level. Using these datasets, we evaluate several deep-learning embedding methods for their suitability to sentence clustering in support of essay grading. We find the suitable method choice to depend on the nature of the exam question and the answers, with deep-learning methods being capable of, but not guaranteeing better performance over simpler methods based on lexical overlap.

Keywords

deep learning, essay clustering, text similarity, paraphrase, grading support

1. INTRODUCTION

Essay-type questions have been shown to help with the retention of learned material [10] but are time- and labour-consuming to evaluate. Computational methods can be used to grade essays, or to assist in their evaluation. Examples of the latter approach include pre-processing to show statistics of student answers such as average answer length and keywords [11], comparing student answers to a given text [11], generating word clouds of student answers [6], and grouping student answers into clusters of similar answers [2]. Most of these systems target the pre-processing and analysis of short answers, and less effort has been dedicated to computer-aided assessment of longer essays. One approach to reducing human effort in fact-based student essay assessment computationally would be to identify similar arguments in student essays. This approach draws inspiration from qualitative research methods where interviews are first transcribed verbatim, and categories are then formed and themes are created [5]. By identifying recurring arguments across a cohort of essays, it is expected that human grading effort could be reduced, much like the analysis of interviews is made simpler after forming categories.

In this paper, we evaluate the applicability of several representative deep learning methods to the task of identifying distinctly-phrased, but semantically near-equivalent segments of student essays[1]. We approach the task from two angles. As an *information retrieval* (IR) problem, whereby given a query text, e.g. a reference answer or an essay, the task is to retrieve the matching essays from the cohort, and establish their mutual correspondence down to sentence level. The other approach is that of *clustering*, where the objective is to discover groups of sentence-long segments with same meaning in the essay cohort. We test several algorithms, including TF-IDF [7], LASER [1], BERT [4], and Sentence-BERT [13]. To evaluate these algorithms, we gather and annotate two sets of factual essays written in exams by Finnish university students.

2. DATASETS

We collected Finnish essays written by bachelor's level students as answers to exam questions. Two sets of essays replying to questions from two courses were selected for manual annotation. The annotator was a PhD student from a different discipline than the domain of the essays. The goal of the annotation was to identify similar arguments in separate essays. The data were annotated by cross-referencing the arguments found in every essay, and assigning textual labels to recurring arguments or concepts on a sentence level. Specifically, all essays were first segmented into sentences, and each sentence was then assigned zero or more textual labels representing its content. If an argument appears more than once, it is given a distinct label which is assigned to all sentences containing that argument. For an argument to be considered recurring, the two sentences are required to clearly aim to communicate the same information about a common subject matter. An example of two sentences that are considered to have the same argument (on the pros and cons of group interviews in research): "It is not the quieter and more timid individuals that come out, but the loudest ones come to the fore." and "In a group interview, there is a danger that some will talk too much and some will not have a turn to speak at all." Both of these sentences describe

[1]We refer readers to `https://arxiv.org/abs/2104.11556` for a more detailed version of the paper

Li-Hsin Chang, Iiro Rastas, Jenna Kanerva, Valtteri Skantsi, Sampo Pyysalo and Filip Ginter "Deep learning for sentence clustering in essay grading support". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 614-618. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Table 1: Dataset statistics

	Research methods	Accounting standards
No. of essays	47	10
Total no. of sentences	486	158
No. of labels	59	34
Avg. no. of labels per sentence	1.29	0.82

the imbalance of expression of opinions in group interviews. In the next example, however, the two sentences are considered to have different arguments, despite both of them being related to the role of trust in interviews. "In interviews, a trusting relationship must be established between the interviewee and the interviewer, which can be challenging." and "If the interviewee remains anonymous, one can also openly discuss more sensitive topics, especially when one is alone with the interviewer." This is because the two sentences make opposing arguments: the former takes a positive perspective towards the role of trust in interviews, while the latter views it as a challenge. Clearly, these communicate different information. For each dataset, the number of labels thus depends on the number of recurring arguments in the essays, and the annotation scheme differs. We estimate that the development of the annotation scheme and the annotation effort required about two person-weeks in total. We note that we do not expect to annotated all sets of essays that are to be evaluated. Instead, these two sets of annotations serve as benchmarks for testing ideas on automatically assisting essay evaluation. The two resulting datasets are introduced below. The key statistics of the two datasets are summarized in Table 1 and the distribution of the labels in the two datasets is illustrated in Figure 1 in the Appendix.

2.1 Research methods dataset

The first dataset is created from student essays from the course "Research process and qualitative research methods" (henceforth *Research methods*). The essays answer the question, "Consider the positive and negative aspects of interviews". Several main points are frequently mentioned by students: for example, almost all students discussed how time consuming interviews can be (label `time_consuming`). 93% of the dataset sentences have at least one label, indicating that the great majority of sentences involve at least one argument repeated in other essays.

2.2 Accounting standards dataset

The second dataset consists of student essays from the course titled "IAS/IFRS accounting standards" (henceforth *Accounting standards*). The essay prompt is "What are the components of IFRS financial statements? Consider the significance of the various components in the light of the qualitative criteria for the financial statement information". The label distribution of this dataset is more even, and almost a third of the sentences do not have a label. This may be due to the fact that there are fewer essays in this dataset. This implies that given one main argument, it is less likely that the argument is also mentioned by somebody else.

3. SENTENCE REPRESENTATIONS

To identify sentences with similar arguments, we consider a set of methods for representing each sentence with a vector, which allows efficient computation of sentence similarity via the similarity of their vectors. As baselines, TF-IDF vectors and average of word embeddings are used for sentence representation. For deep learning methods, the encoders LASER, BERT, and Sentence-BERT are tested. The distance measure used is the cosine similarity between two sentence vectors, a standard metric applied also in previous studies.

3.1 TF-IDF

Term frequency–inverse document frequency (TF-IDF) is a family of popular IR metrics that estimate the importance of a given word in a document from a document collection based on the number of times the word appears in the document (term frequency) and the inverse of the number of documents the word appears in (document frequency) [7]. TF-IDF can be applied with words or character sequences. For this baseline, all the tokens in a sentence are first lemmatized using the Universal Lemmatizer [8]. Character ngrams, specifically bigrams, trigrams, 4-grams and 5-grams, are created out of text inside word boundaries. We note that the TF-IDF encoding generates sparse high-dimensional vectors where there is no inherent similarity between words.

3.2 Average of word embeddings

This baseline represents each sentence using the average of the vector representations of the words in the sentence. We use the Finnish word embeddings created by Kanerva et al. [9] and refer readers to this paper for further details of the embeddings. These embedding were induced using the implementation of the skip-gram algorithm [12] in the `word2vec` software package on Finnish Common Crawl data. The average of word embeddings produces dense, comparatively low-dimensional representations that can capture the similarity between words, but the representation of words is independent of the context they appear in.

3.3 LASER

The Language-Agnostic SEntence Representations (LASER) released by Facebook is a sentence embedding method that aims to achieve universality with respect to language and NLP task. The encoder can encode 93 languages, all of which share a byte-pair encoding [14] vocabulary. The encoder consists of a BiLSTM with max-pooling operation, coupled with an LSTM layer during training on parallel corpora [1]. LASER produces dense, low-dimensional representations that can capture the contextual meaning of words.

3.4 BERT

Bidirectional Encoder Representations from Transformers (BERT) introduced by Google is a deep contextual language representation model [4]. The training objectives of BERT make them cross-encoders, i.e. the model takes in a pair of sentences at a time. However, we encode one sentence at a time and use the mean-pooling of the resulting outputs as the sentence representation. We use the uncased variant of FinBERT, a monolingual Finnish BERT Base model that has been demonstrated to provide better performance in Finnish text processing tasks than multilingual BERT [16]. Like LASER, BERT produces dense, low-dimensional representations that account for context.

Table 2: Results of the IR evaluation

Accounting standards	**Avg First**	**Avg Med**	**Avg Mean**	**Avg Last**	**MRR**	**MAP**
TF-IDF	**4%**	**9%**	**11%**	**24%**	0.47	**0.48**
word2vec	6%	17%	20%	40%	0.47	0.34
LASER	**4%**	13%	15%	33%	**0.53**	0.42
BERT	5%	15%	17%	37%	**0.53**	0.41
SBERT	5%	11%	14%	31%	0.46	0.42
Research methods	**Avg First**	**Avg Med**	**Avg Mean**	**Avg Last**	**MRR**	**MAP**
TF-IDF	**1%**	18%	24%	72%	0.46	**0.28**
word2vec	2%	26%	31%	79%	0.34	0.19
LASER	2%	19%	26%	73%	0.42	0.23
BERT	**1%**	**17%**	23%	70%	**0.49**	**0.28**
SBERT	2%	**17%**	**22%**	**65%**	0.43	**0.28**

3.5 Sentence-BERT

Sentence-BERT (SBERT) trains BERT models using Siamese and/or triplet networks to induce a single-sentence encoder specialized for cosine-similarity comparison [13]. We obtain machine translated versions of the SNLI [3] and MNLI [17] corpora using the English to Finnish Opus-MT model [15]. Finnish SBERT is subsequently trained from FinBERT-base-uncased using these two natural language inference corpora. Specifically, the model is fine-tuned for an epoch with learning rate 2e-5 and batch size of 16, with mean pooling as the pooling method. The representations produced by SBERT are dense, low-dimensional, and context-sensitive, like those of LASER and BERT.

4. EVALUATION

The sentence representations are evaluated from IR and clustering perspectives. Six evaluation metrics are used for the IR approach: two well-known metrics **mean reciprocal rank (MRR)** and **mean average precision (MAP)**, and four metrics tailored to our specific task setting. **average of highest rank (Avg first)**, **average of median rank (Avg med)**, **average of mean rank (Avg mean)**, and **average of lowest rank (Avg last)** measure the rank of the highest, median, mean, and lowest rank of the relevant items respectively, as percentage of the whole (0% first rank, 100% last rank), averaged over all items. These four metrics give more insight into the distribution of the relevant retrievals by measuring where, on average, the first, median, mean, and last relevant items are ranked. Since some sentences have more than one label, sentences with at least one overlapping label are considered relevant retrievals for all metrics.

The clustering evaluation measures how well the clustering induced by the vector embeddings corresponds to the clustering induced by the sentence labels. **Cluster accuracy** is based on the most frequent label of a cluster: for each cluster, the majority label is obtained from the ground truth annotations of the sentences in the cluster. A sentence is considered to be correctly clustered if it has the majority label of its cluster as one of its labels. The number of correctly and incorrectly clustered sentences can then be interpreted as an accuracy percentage. We note that random baseline performance varies drastically between different datasets with this metric, so accuracy values are not directly comparable between datasets. **Adjusted Rand index** and **adjusted mutual information** are established clustering metrics. We use sampling to work around the multi-label nature of the annotations: For each sentence with multiple labels, one label is randomly chosen. Then the clusters are evaluated against these labels with the two metrics. This process is repeated 50 times and the values of the metrics are subsequently averaged. The resulting scores are between -1 and 1, and they are adjusted for chance, so that a random clustering has a score close to zero. We use the agglomerative clustering algorithm with ward linkage. Sentences that have no labels, i.e. containing a unique argument, are each given a unique label for the purposes of the clustering evaluation, effectively each forming one singleton cluster. The resulting true number of clusters (60 for the research methods dataset and 95 for the accounting standards dataset) is the clustering model input.

Table 3: Results of the two clustering evaluation methods. Average adjusted Rand (Avg adj. Rand), Average adjusted mutual information (Avg adj. mutual info.), Cluster accuracy (Clus. acc.), Standard deviation (Std dev).

Accounting standards	**Avg adj. Rand**	**Std dev**	**Avg adj. mutual info.**	**Std dev**	**Clus. acc.**
TF-IDF	**0.31**	0.02	**0.33**	0.02	**73%**
word2vec	0.18	0.02	0.23	0.02	69%
LASER	0.21	0.01	0.27	0.01	72%
BERT	0.21	0.01	0.27	0.02	72%
SBERT	0.28	0.01	**0.33**	0.02	**73%**
Research methods	**Avg adj. Rand**	**Std dev**	**Avg adj. mutual info.**	**Std dev**	**Clus. acc.**
TF-IDF	**0.12**	0.01	0.22	0.01	**55%**
word2vec	0.05	0.00	0.13	0.01	41%
LASER	0.08	0.00	0.17	0.01	46%
BERT	0.11	0.01	**0.23**	0.01	50%
SBERT	0.11	0.01	0.22	0.01	51%

5. RESULTS

The IR evaluation results are shown in Table 2. We find that there is no single method that systematically outperforms the others. Surprisingly, for the accounting standards dataset, the advanced methods fail to outperform the TF-IDF baseline, which achieves the highest results for all metrics except MRR. This indicates that while TF-IDF is not the most competitive in consistently ranking relevant items at the highest ranks, it is able to concentrate relevant items towards higher ranks in general. This is particularly evident for the average of the lasts metric, where TF-IDF scores 7% points higher than the second best performer, SBERT. Here the number 24% indicates that, for the accounting standards dataset, TF-IDF on average ranks all the relevant items within rank 24 out of 100. The high performance of TF-IDF on this dataset may be attributed partly to the essay prompt requiring students to list the correct keywords. The elements of the IFRS financial statements are only so many, and these items cannot be paraphrased. Methods that compare strings directly thus outperform methods that use dense vector representations that approximate their meaning.

The research methods dataset, however, does not have such a strong emphasis on exact keyword matching: there are no fixed numbers of keywords that have to be mentioned in the answers. Rather, the pros and cons of interviews as a research method are described, and thus sentences that

describe the same concept using different words are more likely to occur. On this dataset, considering the retrieval of the first relevant item, both TF-IDF and BERT perform best on the average of the firsts metric, while BERT performs best on the mean reciprocal rank. Since the average of the firsts metric is more lenient on lower rankings of first relevant items, we can infer that BERT performs more consistently on the retrieval of the first relevant item. Overall, BERT-based methods obtain better results, with SBERT in particular outperforming the other methods by 5% points on the retrieval of the last relevant items. BERT and SBERT both obtain the highest results on four out of six metrics.

The results of the clustering evaluation are summarized in Table 3. These results clearly tend towards the TF-IDF baseline, while the word2vec-based approach is the weakest, tallying with the IR evaluation. Of the neural methods, SBERT is particularly strong in the accounting standards dataset, while being in line with BERT in the research methods dataset. Of the two sentence embedding methods, SBERT outperforms LASER in all tests. We are surprised to find that the TF-IDF model seems to be better suited to the clustering objective than the neural methods, and will examine this in future work.

Overall, we find that the comparative ranking of the methods varies strikingly depending on the dataset, evaluation setting, and metric. The dataset dependence may partly be explained by the nature of the arguments made: if the argument is required to contain certain specific terms, TF-IDF can be a very strong method. On the other hand, if the argument involves more abstract concepts that can be expressed in many ways, neural methods may have an advantage over methods that are based on exact string matching. While deep neural methods have led to breakthroughs in many NLP tasks, the gain they show here over the simple TF-IDF baseline is quite small even in the cases where they outperform it. This may indicate challenges specific to the task and domain beyond those we have identified here, and calls for further research into the topic. This includes searching for more suitable encoding methods, improved evaluation methods, and also study of how data should best be annotated to develop methods serving the needs of essay graders.

6. DISCUSSION

Our annotation makes at least two assumptions that call for further investigation: the sentence is the unit of annotation, and the labels are categorical and non-overlapping. Figure 1 shows that approximately 57% and 64% of the sentences in the Accounting standards and Research methods datasets (respectively) have exactly one label. Another 33% and 7% (resp.) of sentences do not have any labels. Since labels are only assigned if a main argument appears more than once, these sentences can be seen as singleton clusters with a label that occurs exactly once. With the current annotation granularity, the annotation is best applicable to cases where each sentence conveys a single main argument. However, the annotation statistics indicate that sentence may not always be the most suitable unit of annotation. These include cases where an argument is made across several sentences, and where a sentence makes several arguments.

In addition to issues related to the sentence as a unit of annotation, there is also a degree of subjectivity to their labeling. For example, in the Research methods dataset, the two labels `workload` and `time_consuming`, which state that interviews are labor-intensive and time-consuming respectively, could arguably be merged. For such boundary decisions to be helpful for essay graders, the marking criteria play a central role and there is no universal cut-off. As an alternative to disjoint categorical labels, one could consider that the arguments (and the labels that represent them) can be organized hierarchically. For instance, in the research methods dataset, the label `interviewer_influence` represents the argument that the stance of the interviewer may affect the research results, and the label `unnatural_performance` describes the affect of the interview situation on the performance of interviewees. On a higher level, both of the labels convey the research results being negatively affected by artificial factors. For these two datasets, the boundary decisions also depend on the sample size: if there are more essays, chances are that a small number of students make the exact same argument, in which case the boundary is unambiguous, or could be seen as a subcluster of a bigger cluster. We hope to address these and related challenges in future work.

One focus of our ongoing work is the practical use of the clusters. An approach to capitalizing on these clusters would be to make them manually adjustable, i.e. examiners can adjust the contents of the clusters, create new clusters, and delete clusters. These clusters can then be color-coded or annotated with text, indicating whether the presence of a certain cluster is desirable in an essay. In addition, if reference answers are available, essays with more overlapping clusters with the reference answers can be automatically identified.

7. CONCLUSIONS AND FUTURE WORK

We focused on the task of computer-assisted assessment of comparatively long essays through the perspectives of IR and clustering. We have created two datasets based on two exam questions from different fields, on which we tested several deep-learning methods with respect to their ability to retrieve and cluster sentences containing the same arguments paraphrased. We found no method to be universally best; rather, the results depend on the nature of the essays under assessment. Overall, the difference between the state-of-the-art deep learning methods and the much simpler TF-IDF baseline is not numerically large, leaving clear room for further development and application of more advanced methods for embedding meaning. Developing such methods, as well as further practical testing of the approach constitute our future work.

8. ACKNOWLEDGMENTS

The research presented in this paper was partially supported by the European Language Grid project through its open call for pilot projects. The European Language Grid project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 825627 (ELG). The research was also supported by the Academy of Finland and the DigiCampus project coordinated by the EXAM consortium. Computational resources were provided by *CSC — the Finnish IT Center for Science*. We thank Kaapo Seppälä and Totti Tuhkanen for administrative support and data collection.

9. REFERENCES

[1] M. Artetxe and H. Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610, 09 2019.

[2] S. Basu, C. Jacobs, and L. Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics*, 1:391–402, 2013.

[3] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[5] C. L. Erlingsson and P. Brysiewicz. A hands-on guide to doing content analysis. In *African journal of emergency medicine : Revue africaine de la medecine d'urgence*, 2017.

[6] S. Jayashankar and R. Sridaran. Superlative model using word cloud for short answers evaluation in elearning. *Education and Information Technologies*, 22:2383–2402, 2016.

[7] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[8] J. Kanerva, F. Ginter, and T. Salakoski. Universal Lemmatizer: A sequence to sequence model for lemmatizing Universal Dependencies treebanks. *Natural Language Engineering*, pages 1–30, 2020.

[9] J. Kanerva, M. Luotolahti, V. Laippala, and F. Ginter. Syntactic N-gram collection from a large-scale corpus of internet Finnish. In *Proceedings of the Sixth International Conference Baltic HLT 2014*, pages 184–191. IOS Press, 2014.

[10] J. D. Karpicke and H. Roediger. The critical importance of retrieval for learning. *Science*, 319 5865:966–8, 2008.

[11] J. McDonald and A. C. M. Moskal. Quantext: Analysing student responses to short-answer questions. *Me, Us, IT*, pages 133–137, 2017.

[12] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[13] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *EMNLP-IJCNLP*, pages 3982–3992, 2019.

[14] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *ACL*, 2016.

[15] J. Tiedemann and S. Thottingal. OPUS-MT — Building open translation services for the World. In *EAMT*, 2020.

[16] A. Virtanen, J. Kanerva, R. Ilo, J. Luoma, J. Luotolahti, T. Salakoski, F. Ginter, and S. Pyysalo. Multilingual is not enough: BERT for Finnish. *arXiv preprint arXiv:1912.07076*, 2019.

[17] A. Williams, N. Nangia, and S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *ACL*, pages 1112–1122, 2018.

APPENDIX

A. LABEL DISTRIBUTION

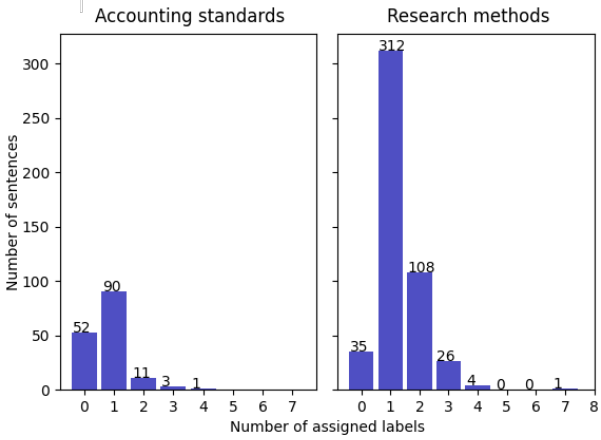


Figure 1: Number of labels per sentence

To Scale or Not to Scale: Comparing Popular Sentiment Analysis Dictionaries on Educational Twitter Data

Conrad Borchers
University of Tübingen
conrad.borchers@student.uni-tuebingen.de

Joshua M. Rosenberg
University of Tennessee, Knoxville
jmrosenberg@utk.edu

Ben Gibbons
Emory University
ben.gibbons@emory.edu

Macy Alana Burchfield
University of Tennessee, Knoxville
mburchf3@vols.utk.edu

Christian Fischer
University of Tübingen
christian.fischer@uni-tuebingen.de

ABSTRACT

The extraction of sentiment from text requires many methodological decisions to make inferences about mood, opinion, and engagement in informal learning contexts. This study compares sentiment software (SentiStrength, LIWC, tidytext, VADER) on $N = 1{,}382{,}493$ tweets in the context of the Next Generation Science Standards reform ($N = 546{,}267$) and U.S. State Educational Twitter Hashtags ($N = 836{,}226$). Automated sentiment classifications were validated on $N = 300$ hand-coded tweets. Additionally, we developed a discrepancy measure to identify tweet features associated with scale inconsistency. Results indicated that binary sentiment classifications (positive/neutral vs. negative) were more accurate than trinary classifications (positive, neutral, negative). Combined tidytext dictionaries and VADER outperformed LIWC for negative sentiment, which was overall difficult to classify reliably while positive sentiment was classified with high accuracy across all four dictionaries. Thus, researchers are encouraged to (a) consider employing overall sentiment scales or positive/neutral to negative ratios based on binary classification to characterize their sample, (b) aggregate multiple dictionaries or use domain-specific sentiment dictionaries, and (c) be aware of the current limitations of detecting negativity through dictionary-based sentiment analysis in educational contexts.

Keywords

Social media data, sentiment analysis, online communities

1. INTRODUCTION

Sentiment analysis extracts positive and negative emotions from text. Its many applications include stock market prediction [22], marketing research [29], and, recently, investigating public sentiment on educational reforms on Twitter [32, 38]. Sentiment analysis typically requires numerous methodological decisions, such as deciding whether to use a dictionary-based or a supervised machine learning approach and determining how sentiment measures are suited to the investigation of a particular domain (e.g., VADER for social media data) [13, 30].

User-defined sentiment dictionaries (UDDs) rely on matches of word occurrences with a value in their dictionary, with little overlap often yielding less valid results. Whereas many sentiment measure validation studies investigate binary (i.e., positive and negative) sentiment classifications [1, 25, 28], less research has systematically compared trinary classifications (i.e., positive, neutral, negative) and sentiment scales. Furthermore, as sentiment classifiers do not generalize well across domains [2], sentiment validation studies are needed to inform educational researchers utilizing the increased availability of big data in education [7]. This study examines the performance of popular sentiment analysis methods in the context of a particular, social media-based data source: large education-related Twitter communities.

The motivation of this study is two-fold. First, sentiment measures can give insight into how and why teachers engage in online communities on Twitter, a potentially novel form of informal teacher learning [6, 33]. Second, public opinion and sentiment can be viewed as a proxy for successful reform implementation [4, 27]. Wang and Fikis applied SentiStrength on more than 660,000 tweets related to the Common Core State Standards, finding sentiment, including that expressed by teachers, to be largely negative [38]. In contrast, Rosenberg et al. found largely positive sentiment in 570,000 NGSS-related tweets through the same SentiStrength algorithm [32]. However, the validity of the utilized sentiment methods was not examined.

2. RESEARCH BACKGROUND

2.1 Sentiment Analysis Methods and Tools

Sentiment analysis is frequently carried out through *user-defined dictionaries* (UDDs) [9]. UDDs contain sets of labeled words that are rated on affect dimensions (e.g., valence, potency, activity) and matched to word occurrences in texts [23]. Researchers can either use pre-defined dictionaries or create their own dictionaries [9]. UDD methods

Conrad Borchers, Joshua Rosenberg, Ben Gibbons, Macy Alana Burchfield and Christian Fischer "To Scale or Not to Scale: Comparing Popular Sentiment Analysis Dictionaries on Educational Twitter Data". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 619-624. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

examine words individually, potentially neglecting figurative language and ambiguous phrases [36]. This study examines four popular examples of UDD software: (a) SentiStrength, (b) Linguistic Inquiry and Word Count (LIWC), (c) the R-package tidytext, and (d) the social-media attuned software VADER.

SentiStrength outputs two truncated five-point scales [37] which is different from many other UDD implementations. It offers feature selection options and measures sentiment weight, which is the intensity or strength of positivity or negativity in a text, as opposed to simply comparing the frequency of sentiments in a text [14].

LIWC is possibly the most popular text analysis software [17]. It uses a well-validated default dictionary [16] and contains around eighty subdictionaries of topics for which it outputs individual scales [26]. LIWC has been used to infer psychological processes and constructs (e.g., emotional expressions) from text [36].

Tidytext [34] does not provide its own default dictionary. At its core, it strives to pre-process input text which is then analyzed through any input dictionary [35]. Tidytext provides functions for converting text into a "one-token-per-document-per-row" format which may ease text analysis.

VADER (Valence Aware Dictionary for Sentiment Reasoning) features multiple subdictionaries and considers word order and degree modifiers (e.g., "very", "slightly", "somewhat") [5]. It performs well in sentiment analyses of social media content (including from Twitter) while remaining applicable to other contexts [5, 13]. That said, we found the R implementation of VADER to take around 80 times longer to compute compared to the other methods.

2.2 Research Questions

This study examined the validity of SentiStrength, LIWC, tidytext, and VADER in the context of educational Twitter data with the following research questions (RQs):

RQ1: How valid are the employed sentiment measures with respect to human coding of sentiment?

RQ2: How discrepant are sentiment scales and are correlations among scales consistent with these discrepancies?

RQ3: Which features of texts (i.e., the number of words, likes, retweets, and context) account for scale discrepancy?

3. METHOD

3.1 Sample

The study utilized tweets related to the Next Generation Science Standards reform and large educational state-wide hashtags (1,382,493 tweets, 156,446 users) posted between July 2008 and October 2020. Search terms included the #NGSSchat hashtag (N = 175,094 tweets, N = 67,060 of which being inside of designated chat-sessions), the terms "ngss" (without #NGSSchat, N = 312,167 tweets) or "next gen[eration] science standard[s]" (N = 59,006 tweets). In addition, we included tweets from 47 State Educational Twitter Hashtags (N = 836,226). Tweets not recognized as of English language by the Twitter API were omitted (5.0%).

3.2 Sentiment Measures

To investigate the validity of different sentiment measures, we used SentiStrength [37], LIWC [26], tidytext [34], and VADER [13] to obtain (a) binary and trinary classifications, (b) unidimensional (positive and negative) sentiment scales, and (c) a bidimensional sentiment scale rating for all tweets. While SentiStrength has binary and trinary classification methods, we subtracted negativity ratings from positivity ratings to obtain overall scores and defined a tweet as neutral if that overall rating was 0 (over 0 as positive, under 0 as negative) for LIWC and tidytext. For tidytext, we used the NRC [24], Loughran-McDonald [20], AFINN [10], and Bing [12] dictionaries, standardizing ratings by the number of words in each tweet and averaging across available ratings. The remaining non-matches were assigned a 0. For VADER, we used its internal compound score as overall scale and classified tweets as neutral if that score was between -0.05 and 0.05 (instead of 0) [13]. Binary classification combined positive and neutral tweets, such that neutral tweets were coded as positive, as done in previous validation studies [8] and since we observed that SentiStrength always classified tweets rated neutral in its trinary method positive in its binary method. Additionally, we defined ambiguity measures for all sentiment dictionaries as the sum of the absolute values of their positivity and negativity ratings.

3.3 Additional Variables

Continuous predictor variables included the number of likes, retweets, and words (excluding links and user mentions) of each tweet. To account for some features of the specific data sets we analyzed, we created a *categorical predictor variable* indicating whether a tweet was from the NGSS or SETHs data set (and, for the NGSS data set, whether the tweet was posted inside of #NGSSchat, designated chat-sessions of #NGSSchat, or included the term "ngss").

3.4 Data Analysis

3.4.1 Hand-coding and validation

To provide a validation set of tweets to investigate how UDDs compare to human-evaluated sentiment (RQ1), two raters hand-coded 300 randomly sampled tweets on two 1-5 scales for positivity and negativity, similar to SentiStrength. Our two raters reached a consensus of $\kappa = 0.728$ for positivity and $\kappa = 0.689$ for negativity after coding 70 tweets independently, fulfilling common thresholds for satisfactory agreement [21]. After discussing and resolving any disagreements, an additional 230 tweets were coded independently. The binary and trinary sentiment classifications of human coders were assigned analogously to how they were created for the other UDDs. We calculated accuracy, precision, recall, and F-score for each category in each classification method (binary and trinary).

3.4.2 Scale consistency and discrepancy index

To quantify scale discrepancy for RQ2, we normalized the sentiment scales to $M = 0$ and $SD = 1$, accounting for SentiStrength's truncation of scales at |5| (contrasting LIWC, tidytext and VADER). As a discrepancy index, we calculated the absolute difference between normalized scales for positivity, negativity, and overall scales for all six pairs of sentiment measures. For each tweet and scale type, the total scale discrepancy was summed up and divided by the

number of comparisons. As a robustness check for our discrepancy measure, we calculated pairwise scale correlations between methods.

3.4.3 Predictive modeling of scale discrepancy

To examine RQ3, we conducted three ordinary least square linear regression models to predict discrepancy in the (a) positivity, (b) negativity, and (c) overall scales through various tweet properties. Model assumptions (normal distribution of residuals, homoscedasticity, linearity assumptions and leverage) were investigated through graphical model tests in R. Robust standard errors (HC3 estimator [19]) were used to address residual heteroscedasticity for discrepancy in the positivity scales. Independent variables included tweet context and the number of words, likes, and retweets of a tweet. We also included binary classifications (0: negative, 1: positive or neutral) to investigate whether scale discrepancies varied with sentiment polarity and ambiguity ratings to estimate whether tweets being high in positivity and negativity were less consistently rated than tweets with less emotional valence. All independent variables had a generalized variance inflation factor (GVIF) of less than 5 [3].

4. RESULTS

4.1 Validation of Sentiment Measures (RQ1)

4.1.1 Dictionary coverage

Coverage characterizes the fit of user-defined dictionaries with the data. Coverage is the relative frequency of texts that had a least one match inside a specific dictionary. We observed a coverage of 58.91% for SentiStrength, 55.7% for LIWC, and 67.7% for VADER. The combined tidytext dictionaries had a coverage of 84.9%. As subdictionaries, coverage was 70.5% for NRC, 62.0% for AFINN, 56.9% for Bing, and 34.9% for the Loughran-McDonald dictionary.

4.1.2 Hand-coded tweets and validation

Comparing human coders with SentiStrength's scale ratings (LIWC, tidytext and VADER do not output 1-5 scales; we describe these later in this section), we found a moderate two-way random effects ICC for absolute agreement [18] for both the positivity scale, $ICC2k = 0.690$ [0.57, 0.77] and the combined, overall scale, $ICC2k = 0.683$ [0.59, 0.75]. The negativity scale exhibited worse agreement, $ICC2k = 0.448$ [0.31, 0.56]. Notably, Cohen's kappa ratings were not satisfactory with $\kappa = 0.301$ for positivity, $\kappa = 0.270$ for the overall scale, and $\kappa = 0.183$ for negativity [21].

Tables 1 and 2 describe the validity of the binary and trinary classifications for SentiStrength, LIWC, tidytext, and VADER. We found trinary classifications to have higher accuracy scores than binary classification (ranging from 85.00% to 88.33% and 56.33% to 67.00%, respectively). Notably, we found classifications of negative tweets to be less accurate than for positive tweets, with F-scores of tidytext and VADER (0.45 and 0.44, respectively) being higher compared to SentiStrength and LIWC (0.38 and 0.29, respectively). To test whether these differences were significant or random, we ran permutation tests with 250,000 simulations [39]. Tidytext and VADER improved compared to LIWC, but not to SentiStrength, (albeit marginally) significantly ($p = .058$ and $p = .047$, respectively), although only 11.67% of tweets were rated as negative by human coders.

Table 1: Binary validation results

SentiStr.			**LIWC**	
Accuracy	85.50		88.33	
	Pos/Neut	**Neg**	**Pos/Neut**	**Neg**
Precision	0.92	0.36	0.90	0.50
Recall	0.91	0.40	0.97	0.20
***F*-Score**	0.91	0.38	0.94	0.29
tidytext			**VADER**	
Accuracy	87.00		88.33	
	Pos/Neut	**Neg**	**Pos/Neut**	**Neg**
Precision	0.93	0.44	0.93	0.50
Recall	0.92	0.46	0.94	0.40
***F*-Score**	0.93	0.45	0.93	0.44

Note: Positive Tweets are either positive *or* neutral in binary classification. Support: 265 Pos/Neut, 35 Neg

Table 2: Trinary validation results

SentiStr.				**LIWC**		
Accuracy	66.00			67.00		
	Pos	**Neut**	**Neg**	**Pos**	**Neut**	**Neg**
Precision	0.66	0.75	0.36	0.65	0.71	0.50
Recall	0.77	0.63	0.40	0.78	0.69	0.20
***F*-Score**	0.71	0.69	0.38	0.71	0.70	0.29
tidytext				**VADER**		
Accuracy	56.33			65.33		
	Pos	**Neut**	**Neg**	**Pos**	**Neut**	**Neg**
Precision	0.50	0.88	0.44	0.59	0.79	0.50
Recall	0.90	0.33	0.46	0.84	0.57	0.40
***F*-Score**	0.64	0.48	0.45	0.69	0.66	0.44

Note: Support: 115 Pos, 150 Neut, 35 Neg

4.2 Consistency of Sentiment (RQ2)

4.2.1 Positivity scale

For positivity scales, LIWC and VADER were the most consistent with each other based on scale correlation ($r = .83$) and mean discrepancy (0.41 *SDs*) followed by tidytext and VADER ($r = .71$, 0.53 *SDs*) and LIWC and tidytext ($r = .63$, 0.61 *SDs*). On average, positivity scales yielded pairwise correlations of $r = .63$ and scale discrepancies of 0.60 *SDs* (Table 3).

4.2.2 Negativity scale

For negativity scales, LIWC and VADER were the most consistent with each other based on scale correlation ($r = .68$) and mean discrepancy (0.33 *SDs*) followed by SentiStrength and LIWC if based on scale correlation ($r = .61$, 1.14 *SDs*) and LIWC and tidytext if based on scale discrepancy ($r = .09$, 0.59 *SDs*). On average, negativity scales yielded pairwise correlations of $r = .35$ and scale discrepancies of 0.83 *SDs* (Table 3).

4.2.3 Overall scale

For overall scales, LIWC and VADER appeared to be closest based on scale correlation ($r = .69$) and mean discrepancy (0.54 *SDs*) followed by LIWC and tidytext ($r = .65$, 0.59 *SDs*), SentiStrength and VADER ($r = .64$, 0.65 *SDs*), and SentiStrength and LIWC ($r = .56$, 0.64 *SDs*), respectively. On average, overall scales yielded pair-wise correlations of $r = .61$ and scale discrepancies of 0.64 *SDs* (Table 3).

Table 3: Pairwise scale correlations (Corr) and discrepancy (Disc) for positivity, negativity, and overall scales of SentiStrength (SS), LIWC (LI), tidytext (TT), and VADER (VA)

	Pos		Neg		Scale	
	Corr	Disc	Corr	Disc	Corr	Disc
SS, LI	0.54	0.64	0.61	1.14	0.56	0.64
LI, TT	0.63	0.61	0.09	0.59	0.65	0.59
SS, TT	0.43	0.78	0.13	1.04	0.52	0.74
SS, VA	0.59	0.66	0.58	1.19	0.64	0.65
LI, VA	0.83	0.41	0.68	0.33	0.69	0.54
TT, VA	0.71	0.53	-.01	0.69	0.60	0.65
∅	0.63	0.60	0.35	0.83	0.61	0.64

Table 4: Linear models predicting aggregated scale discrepancy measures; N = 1,382,493

Predictor	Pos	Neg	Scale
(Intercept)	-0.56***	1.52***	-0.23***
Number of Words	-0.00***	0.01***	0.01***
Number of Likes	0.00	0.00	0.00
Number of Retweets	0.00***	0.00***	0.00***
Context [#NGSSchat]	0.02***	-0.01***	0.03***
Context [SETHs]	-0.01***	0.05***	-0.02***
Context [Chat Hour]	0.00	-0.03***	0.00
Ambiguity [SentiStr.]	0.07***	0.21***	0.07***
Ambiguity [LIWC]	0.11***	0.14***	0.11***
Ambiguity [tidytext]	0.08***	0.19***	0.10***
Ambiguity [VADER]	0.04***	0.06***	0.04***
SentiStr. Binary [1]	0.16***	-0.64***	0.00
LIWC Binary [1]	0.29***	-0.28***	0.30***
tidytext Binary [1]	0.30***	-0.47***	0.11***
VADER Binary [1]	0.14***	-0.48***	-0.03***
R^2	0.22	0.75	0.24

Note: ***p<0.001 **p<0.01 *p<0.05.

4.3 Understanding Scale Discrepancies (RQ3)

Linear models for evaluating scale discrepancies included four notable associations between tweet properties and scale discrepancies (Table 4). First, all four ambiguity measures were positively associated with scale discrepancy measures across all three models, most notably SentiStrength's ambiguity measure with negativity scale discrepancy, $\beta = 0.21$, $t(1382478) = 229.30$, $p < .001$. Second, for binary classifications (i.e., positive/neutral vs. negative), negative tweets tended to have higher negativity discrepancy and vice versa. For example, discrepancy in negativity scales was negatively associated with tweets classified as positive/neutral by SentiStrength, β = -0.64, $t(1382478)$ = -281.13, $p < .001$. Meanwhile, tidytext classifying tweets as positive was associated with increased positivity discrepancy, $\beta = 0.30$, $t(1382478) = 129.71$, $p < .001$. Third, text- and tweet-specific variables (e.g., number of words, likes, and retweets) did not seem to be associated with scale discrepancy, while tweet context had a small effect size. For instance, tweets from State Educational Twitter Hashtags were positively associated with negativity scale discrepancy, $\beta = 0.05$, $t(1382478) = 52.86$, $p < .001$. Fourth, the explained variance in scale discrepancy was highest for negativity scales at 75.3%, followed by overall scales (23.5%) and positivity scales (22.4%).

5. DISCUSSION

5.1 Key Findings

This study evaluates sentiment analysis methods on educational Twitter data. Our three main findings are as follows:

First, negative sentiment is difficult to reliably detect with dictionary approaches. This could be due to nuanced linguistic markers (e.g., sarcasm) that require advanced algorithms to be detected [31]. While this finding aligns with previous work [30], it contrasts initial validations of SentiStrength [37] on a set of around 1,000 MySpace comments [37]. Nonetheless, this highlights the importance of validating commonly used sentiment analysis tools across multiple contexts. Thus, we encourage researchers to carefully examine how negativity may be expressed in their study context.

Second, in the context of educational Twitter data, binary sentiment classifications that combine positive and neutral sentiment are substantially more robust than trinary classifications. Thus, researchers may consider computing the ratio of negative to positive/neutral tweets, similar to a recent Twitter study on the Common Core State Standards [38]. For a continuous variable, our findings suggest using an overall scale, as discrepancy in negativity was substantially associated with ambiguity and binary classifications.

Third, in the context of educational Twitter data, tidytext and VADER produce more accurate classifications of negative sentiment than LIWC. Notably, tidytext also has the highest dictionary coverage. Thus, educational researchers are encouraged to aggregate multiple dictionaries or to create domain-specific sentiment dictionaries for more reliable measures of negative sentiment, for instance, similar to a previous study investigating political expression [11].

5.2 Limitations

This study has two notable limitations. First, the sample size of the training data is relatively small ($N = 300$). That said, it is comparable to sample sizes of previous sentiment measure validation studies [28]. Similarly, the lack of negative tweets in our training data (11.67%) may limit inferences about that particular type of sentiment. The number of negative tweets is considerably smaller compared to previous validation studies utilizing text data from sources such as MySpace, Twitter, BBC forums, and YouTube that include up to 86.84% negative sentiment [8]. Therefore, future validation studies should deliberately sample more negative tweets [13] or sample from contexts with higher expected negativity, such as Common Core State Standards hashtags [38]. Second, this study focuses on dictionary-based sentiment analysis, while future studies might also consider feature extraction and word co-occurrence methods [15].

5.3 Implications

This study highlights the importance of coverage, validity, and scale discrepancy in sentiment analysis, specifically for negative sentiment. For educational Twitter data, this study recommends using binary classifications or overall scales, preferably derived from tidytext or VADER, and encourages replication studies[1] across more educational contexts.

[1]Code: https://github.com/jrosen48/comparing-sentiment

6. REFERENCES

[1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. Sentiment analysis of Twitter data. In *Proceedings of the workshop on language in social media (LSM 2011)*, pages 30–38, 2011.

[2] A. Aue and M. Gamon. Customizing sentiment classifiers to new domains: A case study. In *Proceedings of recent advances in natural language processing (RANLP)*, volume 1, pages 2–1. Citeseer, 2005.

[3] T. A. Craney and J. G. Surles. Model-dependent variance inflation factor cutoff values. *Quality Engineering*, 14(3):391–403, 2002.

[4] A. Edgerton. Learning from standards deviations: Three dimensions for building education policies that last. *American Educational Research Journal*, 57(4):1525–1566, 2020.

[5] S. Elbagir and J. Yang. Twitter sentiment analysis using natural language toolkit and vader sentiment. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 122, page 16, 2019.

[6] C. Fischer, B. Fishman, and S. Y. Schoenebeck. New contexts for professional learning: Analyzing high school science teachers' engagement on Twitter. *AERA Open*, 5(4), 2019.

[7] C. Fischer, Z. Pardos, R. Baker, J. Williams, P. Smyth, R. Yu, S. Slater, R. Baker, and M. Warschauer. Mining big data in education: Affordances and challenges. *Review of Research in Education*, 44(1):130–160, 2020.

[8] P. Gonçalves, M. Araújo, F. Benevenuto, and M. Cha. Comparing and combining sentiment analysis methods. In *Proceedings of the first ACM conference on Online social networks*, pages 27–38, 2013.

[9] J. Grimmer and B. Stewart. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis*, 21(3):267–297, 2013.

[10] L. Hansen, A. Arvidsson, F. Nielsen, E. Colleoni, and M. Etter. Good friends, bad news-affect and virality in twitter. In *Future information technology*, pages 34–43. Springer, 2011.

[11] M. Haselmayer and M. Jenny. Sentiment analysis of political communication: combining a dictionary approach with crowdcoding. *Quality & quantity*, 51(6):2623–2646, 2017.

[12] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

[13] C. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, 2014.

[14] M. Ibrahim. Extracting weight in Twitter SentiStrength dataset to determine sentiment polarity. *Journal of Information Systems Research and Innovation*, 10(3):245–265, 2016.

[15] R. Iliev, M. Dehghani, and E. Sagi. Automated text analysis in psychology: Methods, applications, and future developments. *Language and Cognition*, 7(2):265–290, 2015.

[16] J. Kahn, R. Tobin, A. Massey, and J. Anderson. Measuring emotional expression with the linguistic inquiry and word count. *The American journal of psychology*, pages 263–286, 2007.

[17] M. Kern, G. Park, J. Eichstaedt, A. Schwartz, M. Sap, L. Smith, and L. Ungar. Gaining insights from social media language: Methodologies and challenges. *Psychological methods*, 21(4):507, 2016.

[18] T. K. Koo and M. Y. Li. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of chiropractic medicine*, 15(2):155–163, 2016.

[19] J. S. Long and L. H. Ervin. Using heteroscedasticity consistent standard errors in the linear regression model. *The American Statistician*, 54(3):217–224, 2000.

[20] T. Loughran and B. McDonald. When is a liability not a liability? Textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65, 2011.

[21] M. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282, 2012.

[22] A. Mittal and A. Goel. Stock prediction using Twitter sentiment analysis. *Standford University, CS229*, 15, 2012.

[23] S. Mohammad. Sentiment analysis: Detecting valence, emotions, and other affectual states from text. In *Emotion Measurement*, pages 201–237. Woodhead Publishing, 2016.

[24] S. Mohammad and P. Turney. NRC emotion lexicon. Technical report, National Research Council, Canada, 2013.

[25] I. Mozetič, L. Torgo, V. Cerqueira, and J. Smailović. How to evaluate sentiment classifiers for Twitter time-ordered data? *PloS one*, 13(3):e0194317, 2018.

[26] J. Pennebaker, M. Francis, and R. Booth. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71, 2001.

[27] M. Polikoff, T. Hardaway, J. Marsh, and D. Plank. Who is opposed to Common Core and why? *Educational Researcher*, 45(4):263–266, 2016.

[28] R. Prabowo and M. Thelwall. Sentiment analysis: A combined approach. *Journal of Informetrics*, 3(2):143–157, 2009.

[29] M. Rambocas, J. Gama, et al. Marketing research: The role of sentiment analysis. Technical report, Universidade do Porto, Faculdade de Economia do Porto, 2013.

[30] F. Ribeiro, M. Araújo, P. Gonçalves, M. A. Gonçalves, and F. Benevenuto. Sentibench - a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ Data Science*, 5(1):1–29, 2016.

[31] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 704–714, 2013.

[32] J. Rosenberg, C. Borchers, E. Dyer, D. Anderson, and C. Fischer. Advancing new methods for understanding public sentiment about educational reforms: The case

of Twitter and the Next Generation Science Standards. *OSF Preprints*, 2020.

[33] J. Rosenberg, J. Reid, E. Dyer, M. Koehler, C. Fischer, and T. McKenna. Idle chatter or compelling conversation? the potential of the social media-based #ngsschat network for supporting science education reform efforts. *Journal of Research in Science Teaching*, 57(9):1322–1355, 2019.

[34] J. Silge and D. Robinson. tidytext: Text mining and analysis using tidy data principles in R. *JOSS*, 1(3), 2016.

[35] J. Silge and D. Robinson. *Text mining with R: A tidy approach*. O'Reilly Media, Inc., 2017.

[36] Y. Tausczik and J. Pennebaker. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54, 2010.

[37] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas. Sentiment strength detection in short informal text. *Journal of the American society for information science and technology*, 61(12):2544–2558, 2010.

[38] Y. Wang and D. Fikis. Common Core State Standards on Twitter: Public sentiment and opinion leaders. *Educational Policy*, 33(4):650–683, 2019.

[39] P. Yeh. More accurate tests for the statistical significance of result differences. *arXiv preprint cs/0008005*, 2000.

Knowledge Tracing Models' Predictive Performance when a Student Starts a Skill

Jiayi Zhang, Rohini Das, Ryan S. Baker, Richard Scruggs
University of Pennsylvania
{joycez, rybaker, rscr}@upenn.edu, rohinidas604@gmail.com

ABSTRACT

Previous studies on the accuracy of knowledge tracing models have typically considered the performance of all student actions. However, this practice ignores the difference between students' initial and later attempts on the same skill. To be effective for uses such as mastery learning, a knowledge tracing model should be able to infer student knowledge and performance on a skill after the student has practiced that skill a few times. However, a model's initial performance prediction – on the first attempt at a new skill – has a different meaning. It indicates how successful a model is at inferring student performance on a skill from both their performance on other skills and from the difficulty and other properties of the first item the student encounters. As such, it may be relevant to differentiate prediction in these two contexts when evaluating a knowledge tracing model. In this paper, we describe model performance at a more granular level and examine the consistency of model performance across the number of student instances on a given skill. Results from our research show that much of the difference in performance between classic algorithms such as BKT (Bayesian Knowledge Tracing) and PFA (Performance Factors Analysis), as compared to a modern algorithm such as DKVMN (Dynamic Key-Value Memory Networks), comes down to the first attempts of a skill. Model performance is much more comparable by the time the student reaches their third attempt at a skill. Thus, while there are many benefits to using contemporary knowledge tracing algorithms, they may not be as different as previously thought in terms of mastery learning.

Keywords

Knowledge Tracing, Cold Start, Deep Knowledge Tracing, Bayesian Knowledge Tracing, Performance Factors Analysis, Dynamic Key-Value Memory Networks

1. INTRODUCTION

Knowledge Tracing (KT), attempting to measure student knowledge through performance during learning, is a critical component in modern intelligent tutoring systems and adaptive learning systems [18]. These models use students' previous performance to predict their proficiency on latent knowledge and infer their likelihood of success in future attempts within the learning system.

For well over a decade, Bayesian Knowledge Tracing (BKT; [5]) was the dominant algorithm in research on knowledge tracing – it remains the dominant algorithm in use in systems used at scale by students today. Later on, two waves of competing algorithms emerged – a first wave around 2010, including many psychometrically-influenced algorithms such as Performance Factor Analysis (PFA; [17]) and a second wave in the mid-to-late 2010s based on neural networks, including Deep Knowledge Tracing (DKT; [19]) and Dynamic Key-Value Memory Networks (DKVMN; [26]). Work over the last decade has shown that variants of BKT and PFA that take individual differences and timing into account perform better [9, 15, 25]. The current wave of algorithms based on neural networks, such as DKT and DKVMN, have reported further improvements to model fit [12, 26].

The comparisons between these algorithms have generally focused on metrics comparing overall success at predicting on later items, within the learning system applied to held-out students. In these comparisons, multiple large data sets are typically used, but performance is considered evenly across the data set. However, there are some reasons to think this may be a concerning practice. For one thing, even though the data sets used are typically large, these papers generally do not report if samples are large for all skills. Coetzee [4] notes that BKT parameter estimation is more precise for larger data sets than smaller data sets. Furthermore, Gervet [10] concluded that algorithms based on logistic regression, such as PFA, tend to underfit large datasets, while deep learning based algorithms, like DKT, tend to overfit larger datasets.

More concerningly, many data sets used in student modeling have skills which have only been encountered once or twice by many students, either due to stop-out [3] or rarely-tagged secondary skills. Slater and Baker [22] suggest that BKT models cannot be reliably fit unless there is sufficiently large pool of students who have at least three opportunities to practice each skill. As such, large proportions of existing data sets may reflect a seeming special case. Indeed, accurate prediction on these items likely reflects something different than accurate prediction after a student has had more practice. When a student has not yet worked on a skill, predicting their performance at this point represents what is referred to as a "cold start problem" – needing to perform well before having sufficient data for the current student [24]. It is possible that some more recent algorithms may perform better in these situations than earlier algorithms, either by using information from the student's performance on other skills or information on the difficulty or other properties of specific items. However, this better performance may reflect something different than the student's knowledge of the current skill being studied. As such, it may be meaningful to separate out cold start situations (for a given student and skill) from situations where the model has sufficient data to estimate the current skill by itself, when comparing KT algorithms.

Jiayi Zhang, Rohini Das, Ryan S. Baker and Richard Scruggs "Knowledge Tracing Models' Predictive Performance when a Student Starts a Skill". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 625-629. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

In this paper, we study how the performance of three KT algorithms changes, depending on how much data the algorithms have on the current student's performance on the current skill. We compare the classic algorithms BKT and PFA to a more recent neural network-based algorithm, DKVMN, using the ASSISTments 2009-2010 Skill Builder data [7]. Within each model, the predictive performance, determined by AUC ROC (Area Under the Receiver-Operating Characteristic Curve) and RMSE (Root Mean Square Error) was analyzed at students' first through eighth encounter on a skill, reflecting the changes in model performance as students practice a skill more. We conclude with a discussion of the implications of our finding, for both the evaluation and use of knowledge tracing models.

2. METHODS

2.1 Data

In this study, we utilized the ASSISTments Skill Builder 2009-2010 dataset [7], using the updated version which represents an item requiring multiple skills as a single data point [23]. This specific dataset was chosen because it has clearly defined skills and because this dataset had frequently been used to compare KT models in previous research [11, 13, 14, 23, 27].

In the data preprocessing stage, we removed items not linked to any skill. Each student attempt was annotated with how many opportunities to practice the relevant skill(s) the student had encountered so far – i.e., the first instance means the learner is encountering a skill for the first time, the eighth instance indicates that the learner is encountering the skill for the eighth time. The resultant data set consisted of 4,151 students who attempted 16,891 unique problems on 101 skills, resulting in 274,590 responses. While all the skills were included in model training, only the four most common skills are discussed below (see Table 1).

While using the ASSISTments platform, students have to correctly answer *n* problems in a row to achieve mastery of a skill (where *n* is set by the teacher, but is usually three) and can only then move on to another skill. Given the design of the platform's three-in-a-row mastery learning approach, there is a drop in sample size as the number of instances increases (a common pattern in adaptive learning systems). There is also attrition due to stop-out, where students stop working on a problem set without mastering it [3]. Table 1 shows that across all four skills, the number of students encountering a specific skill *n* times decreased with instance. Of the four skills, an average 20% and 45% attrition rate is observed on the third and eighth instances, respectively.

Table 1: Number of students per instance in each skill

Skill Name	1	2	3	4	5	6	7	8
Addition and Subtraction Fractions	1353	1066	978	920	836	756	692	625
Addition and Subtraction Integers	1226	1021	790	693	640	579	510	460
Conversion of Fractions Decimals & Percents	1225	1145	1121	1034	982	928	852	781
Equation Solving Two or Fewer Steps	961	877	857	821	795	745	722	690

2.2 Model Construction

We constructed the following three knowledge tracing models with the preprocessed ASSISTments 2009 dataset: BKT, PFA and DKVMN. Each model was implemented with 5-fold student-level cross-validation. For the cross-validation, the dataset was split into five folds at the student level. Four folds were used to train the model and the trained model predicted student's performance in the 5th fold. Each part acted as the test set once. Predictions in the test sets were combined and used to compute AUC and RMSE for each opportunity to practice, within each skill. For comparability, the original skills were used to calculate opportunities to practice rather than the new skills derived by DKVMN. The folds were kept the same across models, reducing the likelihood of randomly favoring one algorithm over another. The metrics were averaged across the four skills in each instance for each model.

BKT and PFA predict students' success at each attempt based on their previous performance on the skill. When predicting a student's success on the first attempt of a new skill, without having any prior data, the initial prediction made by BKT and PFA reflect the overall student performance across the entire (training) data set on that skill, instead of the individual student's knowledge level on the skill. By contrast, the deep learning model DKVMN utilizes all of a student's historical data and exploits the underlying relationships between concepts. This transferability of prediction across skills can be expected to give the algorithm an advantage of making the initial predictions on a newly-encountered skill. In fact, [14] studied the effect of interaction among skills in DKT, a closely-related deep learning model, and compared it to BKT. By comparing different approaches to leverage skill data, they concluded that DKT's better performance may be largely due to their use of a student's performance on one skill to predict performance on another skill, whereas skills are strictly separated in BKT. PFA occupies a middle ground, as skills do not directly influence each other, but their combinations in the training set may influence the model parameters found during fitting.

The two widely studied deep learning algorithms DKT and DKVMN utilize neural networks to discover underlying relationships among skills and items when predicting student performance. Because of this, both algorithms have shown significant improvements in model fit compared to traditional algorithms. However, DKT maps the relationships on item level while DKVMN fits a skill model from scratch by considering the relationship among skills and items. Given the purpose of the study is to understand whether transferring information between skills influences a model's accuracy during the first few opportunities, DKVMN is a closer comparison to BKT and PFA within the class of deep learning-based KT algorithms.

2.2.1 Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT; [5]) inputs performance into a simple Markov model that is also a Bayesian Network [20]. To fit BKT, we applied BKT-Brute Force [1] to the data set with a floor of 0.01 for all probabilities and a ceiling of 0.3 for guess and slip to avoid model degeneracy [2]. The algorithm produced estimations for guessing, slipping, initial knowledge, and learning transition probabilities for each of the skills, which were then used to predict the probability of success for each student on each opportunity to practice each skill.

2.2.2 Performance Factors Analysis

Performance Factors Analysis (PFA; [17]) is a model that predicts learner performance using a logistic function that models changes in performance through learners' success and failures within a skill. In this study, following the formulas in [17], the basin hopping algorithm was used to fit the model to obtain the optimal parameters. A set of parameters for success, failure and skill difficulty was derived for each skill, which were then used to compute the

probability P(m) that the student would perform correctly, for each student at each opportunity to practice each skill.

2.2.3 Dynamic Key-Value Memory Networks

Developed based on neural networks, Dynamic Key-Value Memory Networks (DKVMN; [26]) employs two matrices that capture states and the relationships between skill and student mastery to predict performance on items and estimate mastery on a set of automatically-derived skills. We utilized code from Zhang et al. [26] to implement the DKVMN model and used the set of parameters that produced the optimal outcome for the ASSISTments 2009 dataset in the study. The model outputs a probability of success for each student at each problem.

3. RESULTS

3.1 AUC Results

Table 2 summarizes the average AUC results for each of the eight opportunities to practice each skill and the combined AUC for opportunities three through eight in the BKT, PFA, and DKVMN models. Additionally, the overall AUC across the first eight opportunities is also reported for the four skills. Note that the overall AUC only includes the targeted four skills in the first eight attempts and therefore, should not be considered to be the overall AUC of the algorithm across the entire data set.

For the first eight instances, a general upward trend is observed in AUC for all three models. Starting at the first instance, the AUC value for BKT is 0.49, PFA is 0.52, and DKVMN is 0.65. At this point, the AUC value for the DKVMN model is much greater than that of other two models, by approximately 0.15. Compared to BKT and PFA, DKVMN is better at making the initial prediction on the very first time a student sees a skill. In fact, at this point, both BKT and PFA are performing at or below chance.

In the following instances, the values of BKT and PFA became closer to the performance of DKVMN. In fact, by the fourth instance, the models' AUC values were fairly similar, having a range of 0.65-0.70. From the fourth opportunity to the eighth, the AUC values increased by 0.02 to 0.06 across skills. Performance stayed similar between algorithms at this point, but DKVMN still tended to achieve slightly higher performance. Across the 3rd-8th opportunities, DKVMN averaged AUC 0.02-0.05 higher than the other two algorithms (0.70 versus 0.68 for BKT and 0.65 for PFA). These trends can be seen in Figures 1-3.

Table 2: Average AUC values in each instance

Model Type	1	2	3	4	5	6	7	8	3-8	All (1-8)
BKT	0.49	0.63	0.68	0.68	0.68	0.68	0.66	0.70	0.68	0.66
PFA	0.52	0.59	0.63	0.65	0.65	0.65	0.66	0.71	0.65	0.63
DKVMN	0.65	0.68	0.70	0.70	0.70	0.69	0.69	0.72	0.70	0.69

Figure 1: AUC results for BKT model across instances

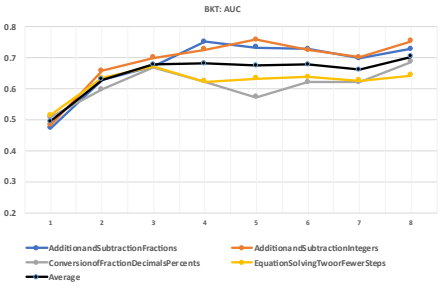


Figure 2: AUC results for PFA model across instances

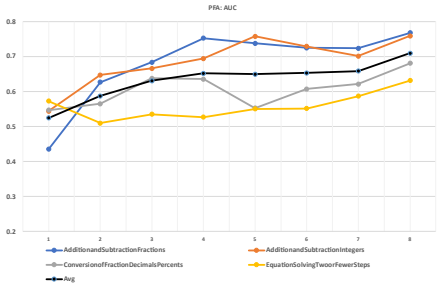
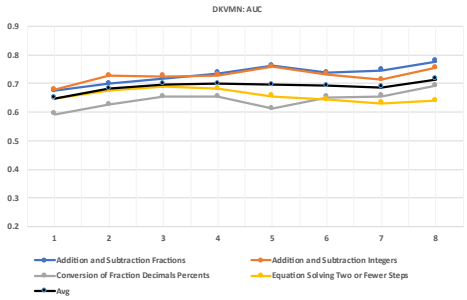


Figure 3: AUC results for DKVMN model across instances



3.2 RMSE Results

Table 3 summarizes the average RMSE results for each opportunity to practice the skills and the combined RMSE for the 3rd-8th opportunities and the 1st-8th opportunities in the BKT, PFA, and DKVMN models. Again, the RMSE reported in the table only considers the targeted four skills in the first eight opportunities.

The RMSE demonstrates a downward trend across the first eight opportunities in all three models. As RMSE measures the difference between actual and predicted values, lower RMSE values indicate more accurate predictions. In the first instance, the RMSE value for BKT is 0.49, PFA is 0.51, and DKVMN is 0.47. As the RMSE value for DKVMN is better than that of BKT and PFA, similar to the AUC value, DKVMN is better able to predict student knowledge at the first attempt (0.02 better than BKT and 0.04 better than PFA).

In the following instances, the values of BKT and PFA became closer to the performance of DKVMN. In fact, by the fourth instance, the models' RMSE values were fairly similar, having a range of 0.43-0.46. From the fourth opportunity to the eighth, the RMSE values in all three models roughly remained the same across skills. Across the 3rd-8th opportunities, DKVMN's average RMSE was similar to BKT and 0.02 lower than PFA (0.44 versus 0.44 and 0.46). These trends can be seen in Figures 4-6.

Table 3: Average RMSE values for all models in each instance

Model Type	1	2	3	4	5	6	7	8	3-8	All (1-8)
BKT	0.49	0.46	0.44	0.44	0.44	0.44	0.45	0.44	0.44	0.45
PFA	0.51	0.48	0.46	0.46	0.47	0.46	0.48	0.46	0.46	0.47
DKVMN	0.47	0.45	0.44	0.43	0.44	0.44	0.45	0.43	0.44	0.45

Figure 4: RMSE results for BKT model in each instance

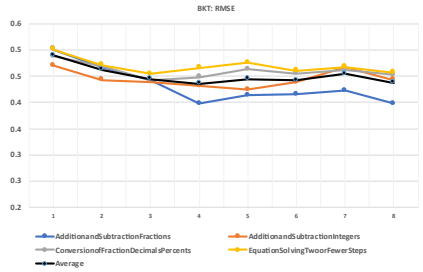


Figure 5: RMSE results for PFA model in each instance

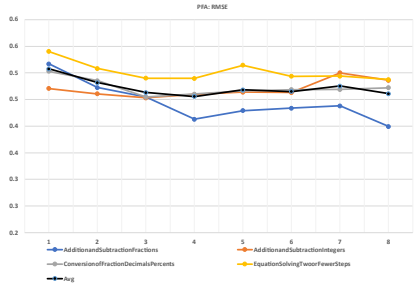
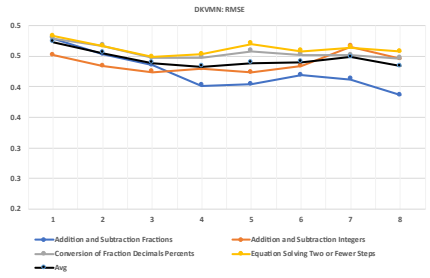


Figure 6: RMSE results for DKVMN model in each instance



4. CONCLUSION AND DISCUSSION

In the last few years, there has been an explosion of interest in new variants to knowledge tracing that achieve higher predictive performance using neural networks. However, this work has generally not yet explored *where* and *when* these algorithms perform better, and what the implications are for using these models in practice. More specifically, previous practices have averaged predictions across students' entire learning history, ignoring the difference between the earliest work and later work on a skill.

In this study, we examined the performance of three KT models, BKT, PFA and DKVMN, across students' history of work on specific skills, and compared how the three models differ in predictive accuracy during the earliest and later opportunities to practice each skill. With all eight opportunities considered together, DKVMN outperformed BKT and PFA in both AUC and RMSE. However, DKVMN's better performance appears to be largely due to its initial prediction on the first attempt on a skill, in which DKVMN 's AUC was 0.16 higher than BKT and 0.13 higher than PFA, and RMSE was 0.02-0.04 better. After the first attempt, BKT and PFA's predictive performance improved substantially, and model performance became closer across the three algorithms after the third attempt, though DKVMN remained slightly better.

The results suggest that much of the difference in performance between these algorithms is due to DKVMN's ability to make more accurate initial predictions by using factors other than mastery of the current skill, such as past performance on other skills and other students' performance on the same item. In other words, a substantial amount of the difference between algorithms appears to be due to factors other than estimating mastery of the current skill the student is working on, from their performance on that skill. This may be especially true in datasets where students stop-out on specific skills [3], or where the skill model is added to or modified after the system is built. In these cases, many student/skill combinations may only occur once or twice, and having relatively higher performance on the first attempt will inflate AUC and RMSE values for models such as DKVMN. This raises the question of what the application is for having better knowledge prediction at the first time when a student sees a new skill. This type of improvement in prediction may be useful to systems that decide which skill a student should work on next (i.e., [6, 28]) but may be less useful in systems that have a predefined order of skills for the student to work on (i.e. [5, 8]) and the student does not move on until they have demonstrated mastery on the current skill.

Given the difference in predictive performance between situations, it may be appropriate to separate out cold start situations (for a given student and skill) from situations where the model has sufficient data to estimate the current skill by itself, when comparing KT algorithms. Specifically, we propose that the calculation of predictive metrics should separate out the predictions on the initial two opportunities to practice each skill from the rest. Adopting this approach will increase our ability to interpret the difference between algorithms and understand how much better a specific algorithm will be for specific use cases.

Two limitations to the current analyses can be addressed in future work. First, our recommendations may not be meaningful for all learning systems where contemporary KT is used. In specific, some systems may not have skill models at all, and may never intend to make inference at the level of interpretable skills. Although these systems typically use an entirely different family of KT models (i.e. [16, 21]), our recommendations would not be relevant in these cases. Second, we have only investigated these issues in the context of a single system and a set of skills for which there is extensive data, and for three algorithms; the generalizability of the findings presented here should be further investigated, using data from other learning systems where, for instance, the granularity of the skills differs. However, only limited effort is needed to separate out practice on early learning opportunities from later learning opportunities when calculating model AUC/RMSE. Therefore, it may be warranted to adopt this approach and see whether practical differences are also found for other contexts and algorithms as well.

Overall, we find initial evidence that one key factor leading to better performance for DKVMN compared to earlier algorithms is its performance in situations, before a student has had significant opportunity to work on a skill. This result leads to recommendations in how to better evaluate KT algorithms and suggests that the benefits of this algorithm may be greater for some applications (deciding which skill a student should work on next) than others (deciding if a student has reached mastery in the current skill they are working on). From the results of this study, future studies conducting research involving KT models may find it useful to calculate performance separately for a student's initial performance and their later performance on a skill; this would provide researchers with more information on how their models are working, and where their greatest benefits and potential are.

5. REFERENCES

[1] Baker, R.S.J. d. et al. 2010. Contextual Slip and Prediction of Student Performance after Use of an Intelligent Tutor. *User Modeling, Adaptation, and Personalization* (Berlin, Heidelberg, 2010), 52–63.

[2] Baker, R.S.J.D. et al. 2008. More Accurate Student Modeling through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. *Intelligent Tutoring Systems* (Berlin, Heidelberg, 2008), 406–415.

[3] Botelho, A.F. et al. 2019. Refusing to try: Characterizing early stopout on student assignments. *Proceedings of the 9th International Conference on Learning Analytics & Knowledge* (New York, NY, USA, Mar. 2019), 391–400.

[4] Coetzee, D. 2014. Choosing sample size for knowledge tracing models. *CEUR Workshop Proceedings* (2014), 117–121.

[5] Corbett, A.T. and Anderson, J.R. 1995. Knowledge Tracing : Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction*. (1995), 253–278.

[6] Craig, S.D. et al. 2013. The impact of a technology-based mathematics after-school program using ALEKS on student's knowledge and behaviors. *Computers and Education*. 68, (2013), 495–504.

[7] Feng, M. et al. 2009. Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*. 19, 3 (2009), 243–266.

[8] Feng, M. et al. 2009. Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*. 19, 3 (Aug. 2009), 243–266.

[9] Galyardt, A. and Goldin, I. 2015. Move your lamp post: Recent data reflects learner knowledge better than older data. *Journal of Educational Data Mining*. 7, 2 (2015), 83–108.

[10] Gervet, T. et al. 2020. When is Deep Learning the Best Approach to Knowledge Tracing? *Journal of Educational Data Mining*. 12, 3 (2020), 31–54.

[11] Gong, Y. et al. 2010. Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting Procedures. *Intelligent Tutoring Systems* (Berlin, Heidelberg, 2010), 35–44.

[12] Khajah, M. et al. 2016. How deep is knowledge tracing? *Proceedings of the 9th International Conference on Educational Data Mining, EDM 2016* (2016), 94–101.

[13] Minn, S. et al. 2018. Deep Knowledge Tracing and Dynamic Student Classification for Knowledge Tracing. *2018 IEEE International Conference on Data Mining (ICDM)* (Singapore, Nov. 2018), 1182–1187.

[14] Montero, S. et al. 2018. Does deep knowledge tracing model interactions among skills? *Proceedings of the 11th International Conference on Educational Data Mining* (2018).

[15] Pardos, Z.A. and Heffernan, N.T. 2010. Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. *In International Conference on User Modeling, Adaptation, and Personalization* (2010), 255–266.

[16] Pavlik, P. et al. 2008. Using Optimally Selected Drill Practice to Train Basic Facts. *Intelligent Tutoring Systems* (Berlin, Heidelberg, 2008), 593–602.

[17] Pavlik, P.I. et al. 2009. Performance Factors Analysis – A New Alternative to Knowledge Tracing. *Proceedings of the 14th International Conference on Artificial Intelligence in Education* (Brighton, England, 2009), 531–538.

[18] Pelánek, R. 2017. Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*. 27, 3–5 (2017), 313–350.

[19] Piech, C. et al. 2015. Deep knowledge tracing. *Advances in Neural Information Processing Systems*. 2015-Janua, (2015), 505–513.

[20] Reye, J. 2004. Student Modelling based on Belief Networks. *International Journal of Artificial Intelligence in Education,*. 14, (1) (2004), 63–96.

[21] Settles, B. and Meeder, B. 2016. A trainable spaced repetition model for language learning. *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*. 4, (2016), 1848–1858.

[22] Slater, S. and Baker, R.S. 2018. Degree of error in Bayesian knowledge tracing estimates from differences in sample sizes. *Behaviormetrika*. 45, 2 (Oct. 2018), 475–493.

[23] Xiong, X. et al. 2016. Going Deeper with Deep Knowledge Tracing. *Proceedings of the 9th International Conference on Educational Data Mining*. (2016), 545–550.

[24] Yang, T.-Y. et al. 2019. Active Learning for Student Affect Detection. *Proceedings of The 12th International Conference on Educational Data Mining* (2019), 208–217.

[25] Yudelson, M. V. et al. 2013. Individualized bayesian knowledge tracing models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 7926 LNAI, (2013), 171–180.

[26] Zhang, J. et al. 2017. Dynamic key-value memory networks for knowledge tracing. *26th International World Wide Web Conference, WWW 2017* (2017), 765–774.

[27] Zhang, J. et al. 2017. Dynamic Key-Value Memory Networks for Knowledge Tracing. *Proceedings of the 26th International Conference on World Wide Web* (Perth Australia, Apr. 2017), 765–774.

[28] Zou, X. et al. 2019. Towards Helping Teachers Select Optimal Content for Students. *International Conference on Artificial Intelligence in Education* (Cham, 2019), 413–417.

Analysis of stopping criteria for *Bayesian Adaptive Mastery Assessment*

Androniki Sapountzi
Vrije Universiteit Amsterdam
Faculty of Behavioral and Movement Sciences
a.sapountzi@vu.nl

Sandjai Bhulai
Vrije Universiteit Amsterdam
Faculty of Sciences
Department of Mathematics
s.bhulai@vu.nl

Ilja Cornelisz
Vrije Universiteit Amsterdam
Faculty of Behavioral and Movement Sciences
i.cornelisz@vu.nl

Chris van Klaveren
Vrije Universiteit Amsterdam
Faculty of Behavioral and Movement Sciences
c.p.b.j.van.klaveren@vu.nl

ABSTRACT

Computer-based learning environments offer the potential for automatic adaptive assessments of student knowledge and personalized instructional policies. In prior work, we introduced an individualized Bayesian model to dynamically assess student's knowledge, based on observed response times and response accuracy. In this paper, we leverage that model as a stopping instructional policy to determine when to stop the assessment. We evaluate several criteria based on the change of performance measures as questions are presented. These include the mean assessment level and the Kullback-Leibler divergence. Student performances are simulated considering their sensitivity to the prior belief for mastery over different educational cases. Our results indicate which criteria offer an efficient assessment, a confident assessment, and which can effectively handle wheel-spinning students.

Keywords

Bayesian Adaptive Mastery Assessment; stopping policy; individualization; empirical analysis; performance model; mastery criteria

1. INTRODUCTION

In adaptive learning systems, mastery is measured as a student performs a skill and demonstrates knowledge by solving a sequence of questions that tap that skill. Learner models that rely on the mastery learning theory are widely used in various personalized adaptive learning systems to infer student mastery sequentially.

In a mastery learning framework, 'under-practicing' and 'over-practicing' are two common pitfalls that cause students to face a practicing or testing burden rather than focus on the skill of their level [1, 2]. This might cause demotivation and low engagement [3, 4, 2, 1, 5, 6]. Particularly, students trapped in a mastery assessment cycle are referred to as wheel-spinning students [7, 3, 8, 9]. They are consistently unable to reach the mastery-success criterion set for the skill, which triggers the system to present even more items. In our previous paper, we proposed,Bayesian Adaptive Mastery Assessment(BAMA), a framework we created to assess a student individually on a single skill given an explicit mean success criterion. From an educational perspective, it can be used as a criterion-referenced assessment to assure mastery [10, 6, 11, 12, 13]. We evaluated the utility function of BAMA as a when-mastery-is-attained policy and show that it accurately recovers the true mastery efficiently, i.e., with few responses. However, this strategy is not sufficient as it assumes that all students at some point will reach that criterion [7, 2, 3, 4, 14, 9].

In this paper, we thereby evaluate the impact of the utility function of BAMA as a stopping policy. We design implicit stopping criteria and we provide an empirical analysis considering the variance of length practice across simulated student performances. We demonstrate that the developed policy delivers meaningful results and identifies any student profile, including wheel-spinners.

2. RELATED WORK

Student profiles aim to portray the individual performance of each learner. Based on the response time of student performances, learning sciences distinguish between struggling fluent from fluent as the latter provide correct responses with short response times [15]. An individual who has not yet acquired the skill and will not demonstrate successful performance is commonly modeled as having a low probability of a correct response [8, 2, 7, 9, 16]. These students are termed as wheel-spinning students [7, 3, 8, 9] and have been linked with long response times [8].

An instructional policy, also known as a stopping policy, refers to the total length of the assessment when a prespecified stopping criterion accompanies the model. The criteria are divided into two categories: (i) an explicit threshold

Androniki Sapountzi, Sandjai Bhulai, I. Cornelisz and Chris Van Klaveren "Analysis of stopping criteria for Bayesian Adaptive Mastery Assessment". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 630-634. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

set to a statistic of the mastery estimator, known as a mastery success criterion, and (ii) an implicit threshold set to the size of change of a statistic of the mastery estimator. The former framework is typically referred to as when-mastery-is-attained policy, and the latter as a when-to-stop policy, as it stops, independent of whether the student has mastered the skill [7].

Substantial research efforts have focused on the impact of a learner model concerning the total number of questions it administers. Machine learning models were designed to detect wheel spinner performance [9]. Frameworks of instructional policies [7, 14] and metrics [5] were proposed for an evaluation of well-known prediction models on the final proposed length. Other work specified a framework for a conceptual interpretation over the stopping criterion [3]. Typically, these models assume a homogeneous class of students and they consider solely response accuracy. Previous research has shown that individualized models lead to significantly different policies [4] and highlighted the importance of response times in stopping policies [9, 12, 17, 18, 13, 10, 19, 2].

3. MODEL AND STOPPING POLICY

Below we briefly discuss the assessment model, the stopping criteria we consider, and the steps of our experiment.

3.1 Bayesian Adaptive Mastery Assessment

In the BAMA model, a student has a constant mastery level Z on a single skill which is the product of two independent random variables, the response time $T \sim \text{Exponential}(\lambda)$ and the accuracy $P \sim \text{Bernoulli}(\theta)$. We denote with τ the maximum response time. The score Z is close to 1 when a student answers correctly and relatively fast with respect to τ. The value of Z becomes zero when a student answers incorrectly, or when the response time exceeds τ. That is operationalized as follows: $Z = P \cdot \left(1 - \frac{T}{\tau}\right)^{+}$.

To keep the formulation tractable, we adopt a Bayesian approach to estimate the true unknown parameters θ and λ of a student. We model θ by a Beta(α, β) distribution, and λ by a Gamma(n, γ) distribution. This represents the prior distribution over the unknown parameters (θ, λ), denoted as p_0, as an initial belief over a student's mastery. The model updates the belief on a posterior distribution p over these parameters under the Bayes rule. As more responses become available, the posterior distributions of the accuracy (the Beta distribution) and the response time (the Gamma distribution) become more centered and peaked around the true values of θ and λ. However, this information is not known in practice and needs to be estimated from the observations received over the assessment.

3.2 Stopping Criteria

A respective policy is concerned with the nature of the estimated Z-score and adopts a different stopping rule. We employ the change of a point estimate, and the change of the distribution. These are computed according to the change observed between consecutive pairs of responses over the sequence. For the analysis and the evaluation of a policy, the following four properties are typically considered [20, 7]: 1) number of administered items, 2) number of non-stopping situations, 3) accuracy with regard to the true value, 4) uncertainty of the experiment and of the model.

The derivative-based stopping rule considers the reduction of changes observed between consecutive pairs of responses as measured with a pre-specified sample statistic of the Z distribution. To put this formally, let $\Delta f_i = f_{i-1} - f_i$ for any function f. Then, our policy proposes to stop after response i when the following decision rule holds.

$$|\Delta h_{i-1}| < \epsilon \wedge |\Delta h_i| < \epsilon, \tag{1}$$

where h_i denotes the value of a sample statistic of the distribution Z after the i-th observation, such as the mean, variance, or any other function. The rule indicates that in a sequence of three responses so far, two values for that rule are computed. Similar to all implicit-based stopping rules, the threshold value denoted as ϵ will also inevitably affect the length of the assessment, i.e., as ϵ gets smaller, the longer the assessment becomes. That is a special case of the probabilistic stopping rule proposed in [7] which doesn't directly generalize to our model.

In our first experiment, we leverage the derivative-based rule by considering the change of the posterior mean from the prior mean. Point-based estimates from sample statistics are all informative metrics that can be employed. However, other estimated statistics may exist to describe the information of a distributional score that may better accommodate a balanced length assessment. Thereby, a more elegant solution would be to calculate a metric that considers the whole distributional information obtained for Z at once.

We compute the second rule based on the reduction of divergence between two consecutive distributions of responses, the starting prior Z_{i-1} and the updating posterior Z_i, after item i has been administered. We formulate this with the Kullback-Leibler (KL) divergence D_{KL} as follows:

$$D_{\text{KL}}(Z_{i-1} \parallel Z_i) = \int_0^1 z_{i-1}(x) \log\left(\frac{z_{i-1}(x)}{z_i(x)}\right) \mathrm{d}x. \tag{2}$$

The quantity $z_i(x)$ describes the density of the distribution Z_i at response i evaluated at x.

3.3 Simulated performance profiles

A student is characterized by the pair (θ, λ) for their performance. For the exposition of our purpose, we take four equidistant intervals of Z defined as: mastered or fluent ($Z \in [0.75-0.95]$), accurate or struggling fluent ($Z \in [0.5-0.74]$), undetermined or average ($Z \in [0.2-0.49]$), wheel-spinning ($Z \in [0-0.19]$). Then, we arbitrarily draw a specific pair of (θ, λ) corresponding to the Z score from each interval. Particularly, we illustrate the following levels: mastered with high accuracy and short response times ($\theta = 0.9, \lambda = 1$) $\rightarrow$ $Z = 0.85$, accurate with high accuracy and long response times ($\theta = 0.9, \lambda = 0.1$) $\rightarrow$ $Z = 0.50$, undetermined with ($\theta = 0.5, \lambda = 0.5$) $\rightarrow$ $Z = 0.46$, and wheel-spinning with ($\theta = 0.1, \lambda = 0.1$) $\rightarrow$ $Z = 0.08$.

4. RESULTS

We evaluate our stopping criteria through simulated student performances. In practice, this translates to n observations of responses $x_1, \ldots, x_n$ according to the student profile (θ, λ)

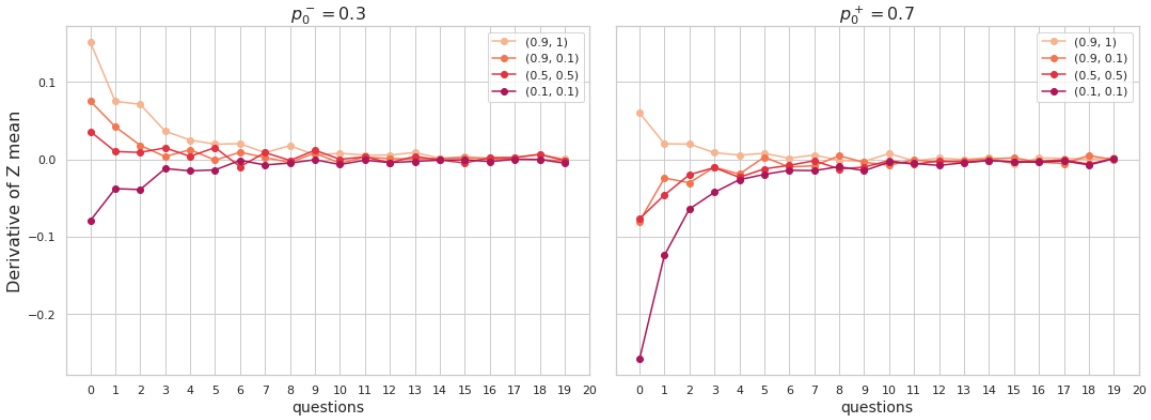


Figure 1: The size of the change between consecutive estimated expected values of Z for a prior $p_0^- = 0.3$ and a prior $p_0^+ = 0.7$.

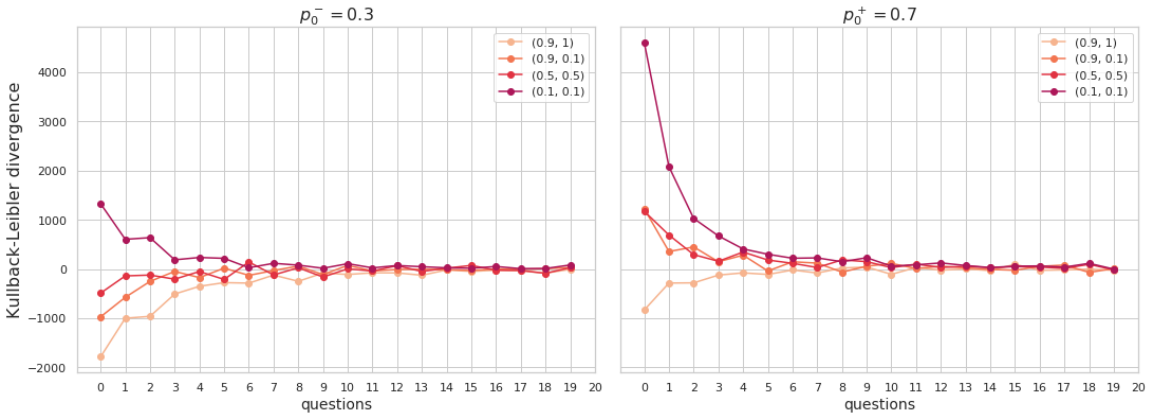


Figure 2: The KL divergence between consecutive estimated distributional scores for a prior $p_0^- = 0.3$ and a prior $p_0^+ = 0.7$.

and prior p_0. We update the prior distribution as observations arrive, i.e., p_i based on p_{i-1} and x_{i-1}. This allows us to collect statistics on the Z-score for each administered question. We repeat this 1,000 times to get accurate results for the statistics. To ensure that the practice length is not highly sensitive to the choice of the prior belief p_0, we simultaneously consider two priors. An optimistic view, denoted as p_0^+, assuming a student who has mastered the skill, and a pessimistic view denoted as p_0^-, assuming a student who has not yet mastered the skill. Considering a fixed maximum number of responses n and updating simultaneously p_0^+ and p_0^- additionally balances the efficiency and certainty of the assessment.

We perform our experiments according to the above procedure for each student profile (θ, λ) and prior p_0 for a sequence of length $n = 20$, similarly to previous research [7, 3, 9]. We set symmetric values of priors as $p_0^+ = 0.7$ and $p_0^- = 0.3$. The value of the maximum permitted response time is arbitrarily set to $\tau = 20$, and the fastest answer to $\lambda = 1$.

4.1 Change of the posterior predictive mean

Figure 1 shows the derivative rule described in Equation (1) implemented for the posterior mean $\hat{\mu}$. Particularly, the magnitude of change $\Delta\hat{\mu}_i$ is depicted over consecutive responses i across the student profiles (θ, λ). The response interval at which $\Delta\hat{\mu}_i$ does not change anymore is observed by the converging lines.

Intuitively, one would expect that the algorithm would propose more questions to wheel-spinning and undetermined students compared to mastered students. However, that is not the case when our starting belief, $p_0^- = 0.3$, is closer to the true posterior. Instead, the mastered students will be proposed to provide more responses. The situation is reversed when we start with an optimistic prior p_0^+.

Second, the algorithm adjusts quickly to the student's practice despite the presence of a non-representative prior. To illustrate this, take the mastered student. Also, take the same length of items, e.g., the first three questions. When we start with a representative prior for the student, in this case p_0^+, the reduction of the change will be twice smaller compared to the reduction of the change observed when we start with the non-representative prior, p_0^-.

4.2 Statistical divergence between consecutive distributional scores

Figure 2 shows the divergence of the estimated distribution $D_{\mathrm{KL}}(i)$ described in (2). Wheel-spinners have $D_{\mathrm{KL}}(i) \geq 0$,

Table 1: Analysis and evaluation of stopping policies per profile, criterion, prior and threshold.

Assessment length: (SE, $\hat{\sigma}$, $\lvert\frac{\mu-\hat{\mu}}{\mu}\rvert$%)				
Stopping rule	Mastered	Accurate	Undetermined	Wheel-spin
$\Delta\hat{\mu}_{.01}^{p_0^+}$	5: (0.0, 0.2, 2.84)	7: (0.0, 0.33, 8.67)	8: (0.01, 0.4, 16.76)	12: (0.0, 0.27, 135.04)
$\boldsymbol{\Delta\hat{\mu}_{.02}^{\mathbf{p_0^+}}}$	4: (0.0, 0.19, 3.49)	5: (0.0, 0.32, 9.99)	4:(0.01, 0.36, 26.62)	7: (0.0, 0.3, 217.43)
$\Delta\hat{\mu}_{.01}^{p_0^-}$	11: (0.0, 0.31, 11.04)	7: (0.0, 0.35, 5.43)	8: (0.01, 0.4, 6.93)	8:(0.0, 0.23, 81.34)
$\Delta\hat{\mu}_{.02}^{p_0}$	9: (0.0, 032, 12.66)	4: (0.01, 0.35, 9.55)	3: (0.01, 0.36, 14.4)	5:(0.0, 0.25, 119.89)
$\Delta D_{\mathrm{KL}}{}_{.02}^{p_0^+}$	6: (0.0, 0.2, 1.92)	12: (0.0, 0.34, 5.76)	8: (0.01, 0.4, 16.76)	7: (0, 0.3, 217.43)
$\boldsymbol{\Delta D_{\mathrm{KL}}}_{\mathbf{.05}}^{\mathbf{p_0^+}}$	4: (0.0, 0.19, 3.49)	8: (0.0, 0.33, 7.11)	5: (0.01, 0.38, 21.53)	6: (0.0, 0.31, 242.25)
$\Delta D_{\mathrm{KL}}{}_{.02}^{p_0^-}$	12: (0.0, 0.31, 10.39)	14: (0.0, 0.35, 4.1)	*19: (0.0, 0.43, 2.56)*	6: (0.0, 0.24, 95.95)
$\boldsymbol{\Delta D_{\mathrm{KL}}}_{\mathbf{.05}}^{\mathbf{p_0^-}}$	7: (0.0, 0.32, 15.77)	8: (0.0, 0.35, 4.94)	4: (0.01, 0.38, 11.09)	6: (0.0, 0.24, 95.95)

in contrast to the mastered students, who have $D_{\mathrm{KL}}(i) \leq 0$. This can be attributed to the prior under- or overestimating the Z score.

The results of $D_{\mathrm{KL}}(i)$ are consistent to the posterior mean $\hat{\mu}$. We observe a shorter length between two responses when the prior is representative for the posterior.

4.3 Analysis and evaluation of the policies

Table 1 reports the results of the implemented stopping policies. For each student profile and stopping rule, as presented by the columns and rows, we find the number of items at which each rule proposes to stop and the variance of the assessment length for different profiles. For each stopping criterion, the prior distribution p_0 is depicted as a superscript and the threshold ϵ as a subscript.

For $\Delta\hat{\mu}$ and the optimistic prior, the simulated students need to solve at most 5-12 questions; whereas for $\Delta\hat{\mu}$ and the pessimistic prior, the simulated students need to solve at most 3-11 questions, depending on the chosen threshold. Considering a single prior, the optimistic one performs better across all students compared to the pessimistic one. Those policies are depicted in bold letters.

For ΔD_{KL} and the optimistic prior, the simulated students need to solve at most 4-12 questions, depending on the threshold value. For ΔD_{KL} and the pessimistic prior, there is a chance of a non-convergent policy. That holds for the undetermined student as the policy converges only at the end. This is depicted with the italic letters in the table.

The assessment length is short when the prior is close to reality. This is depicted for the lenient threshold, e.g., in the case of p_0^+ for a mastered student and p_0^- for an undetermined student. Therefore, we satisfy both priors simultaneously. In that case, the maximum number of questions is 9 for $\Delta\hat{\mu}$. We get the same estimate of items with almost the same uncertainty for both thresholds. Hence, we argue that a shorter assessment length is preferred. It also shows that the policy is less dependent on the value of ϵ.

The results of the lenient threshold stopping policies of ΔD_{KL} and the $\Delta D_{\hat{\mu}}$ show that we achieve an efficient assessment for both priors across all student performances. The satisfaction of both priors is an efficient length considering that in criterion-referenced assessments, at least $n = 4$ responses are required to estimate the mastery of a single skill. Furthermore, we observe that using both priors results in more efficient assessment of wheel-spinning students. In the environment we have simulated, we see that one metric is preferred towards the other under a certain objective. To achieve efficiency for mastered and wheel-spinners, the KL can be used. When the objective is shifted towards efficiency among the average profiles, then the mean could be a more appropriate metric. That doesn't generalize to other settings.

5. CONCLUSIONS

To conclude, we analyzed the performance of different stopping policy rules for the utility function of the BAMA framework. The stopping policy is constructed using both the pessimistic and the optimistic prior for the assessment with a maximum length of $n = 20$. This has several advantages: fluent students will be picked up by the optimistic prior, wheel-spinners by the pessimistic prior, and the other two profiles by either one of the prior distributions. Consistent behavior was found between the two criteria. Furthermore, the lenient threshold is favored in both criteria. The mean assessed mastery level (i.e., $\Delta\hat{\mu}$) stopping criterion slightly outperformed the divergence of assessed mastery level (i.e., ΔD_{KL}). The evaluation of the stopping policies is based on these properties – fewer items, none non-convergent performance case, and relative percentage approximation error is low with high certainty. The simulated data has features that we modelled explicitly. As future work, we plan to evaluate the stopping policies in real-world scenarios with real data and provide a way to represent the average response time and the average response accuracy of the student performance.

6. REFERENCES

[1] E. Joseph, Engagement tracing: using response times to model student disengagement, Artificial intelligence in education: Supporting learning through intelligent and socially informed technology 125 (2005) 88.

[2] R. Pelánek, Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques, User Modeling and User-Adapted Interaction 27 (3-5) (2017) 313–350.

[3] R. Pelánek, Conceptual issues in mastery criteria: Differentiating uncertainty and degrees of knowledge,

in: International Conference on Artificial Intelligence in Education, Springer, 2018, pp. 450–461.
[4] J. I. Lee, E. Brunskill, The impact on individualizing student models on necessary practice opportunities, International educational data mining society (2012).
[5] J. P. González-Brenes, Y. Huang, " your model is predictive–but is it useful?" theoretical and empirical considerations of a new paradigm for adaptive tutoring evaluation., International Educational Data Mining Society (2015).
[6] B. S. Bloom, Time and learning., American psychologist 29 (9) (1974) 682.
[7] T. Käser, S. Klingler, M. Gross, When to stop?: towards universal instructional policies, in: Proceedings of the Sixth International Conference on Learning Analytics & Knowledge, ACM, 2016, pp. 289–298.
[8] Z. Aghajari, D. S. Unal, M. E. Unal, L. Gómez, E. Walker, Decomposition of response time to give better prediction of children's reading comprehension., International Educational Data Mining Society (2020).
[9] Y. Gong, J. E. Beck, Towards detecting wheel-spinning: Future failure in mastery learning, in: Proceedings of the Second (2015) ACM Conference on Learning@ Scale, ACM, 2015, pp. 67–74.
[10] J. P. Lalley, J. R. Gentile, Classroom assessment and grading to assure mastery, Theory Into Practice 48 (1) (2009) 28–35.
[11] M. Bulger, Personalized learning: The conversations we're not having, Data and Society 22 (1) (2016).
[12] H. Peng, S. Ma, J. M. Spector, Personalized adaptive learning: an emerging pedagogical approach enabled by a smart learning environment, Smart Learning Environments 6 (1) (2019) 9.
[13] J. R. Anderson, A. T. Corbett, K. R. Koedinger, R. Pelletier, Cognitive tutors: Lessons learned, The journal of the learning sciences 4 (2) (1995) 167–207.
[14] J. Rollinson, E. Brunskill, From predictive models to instructional policies, International Educational Data Mining Society (2015).
[15] C. Binder, E. Haughton, B. Bateman, Fluency: Achieving true mastery in the learning process, Professional Papers in special education (2002) 2–20.
[16] W. J. González-Espada, D. W. Bullock, Innovative applications of classroom response systems: Investigating students' item response times in relation to final course grade, gender, general point average, and high school act scores, Electronic Journal for the Integration of Technology in Education 6 (2007) 97–108.
[17] C. Lin, S. Shen, M. Chi, Incorporating student response time and tutor instructional interventions into student modeling, in: Proceedings of the 2016 Conference on user modeling adaptation and personalization, 2016, pp. 157–161.
[18] R. Ellis, Measuring implicit and explicit knowledge of a second language: A psychometric study, Studies in second language acquisition 27 (2) (2005) 141–172.
[19] P. De Boeck, M. Jeon, An overview of models for response times and processes in cognitive tests, Frontiers in psychology 10 (2019) 102.
[20] R. E. Stafford, C. R. Runyon, J. M. Casabianca, B. G. Dodd, Comparing computer adaptive testing stopping rules under the generalized partial-credit model, Behavior research methods 51 (3) (2019) 1305–1320.

Detecting Careless Responding to Assessment Items in a Virtual Learning Environment Using Person-fit Indices and Random Forest

Sanaz Nazari, Walter L. Leite, and A. Corinne Huggins-Manley
University of Florida
sanaznazari@ufl.edu

ABSTRACT

Careless responding and keeping students motivated for different tests have been common problems in many areas, especially in education. This study's objective was to demonstrate a novel approach to detect careless responding using person-fit indices developed within the field of psychometrics combined with a random forest. The data used was obtained from various tests in the Math Nation virtual learning platform. The result of person-fit indices as previously used measures of careless responding as well as the result of a random forest classifier to capture careless responding were compared by Receiver Operating characteristic (ROC) analysis and the area under the curve (AUC). The result showed that random forest combined with person-fit indices outperformed person-fit indices directly in detecting careless responding. Some important applications of this method for applied researchers are discussed in the conclusion section.

Keywords

Careless responding, Person-fit index, Random forest classifier, Virtual learning environment

1. INTRODUCTION

Paulhus (1991) defined response bias as "a systematic tendency to respond to a range of questionnaire items on some basis other than the specific item content." Response biases are particularly important as they become a threat to the validity of conclusions and lead to measurement error (e.g., Meade & Craig, 2012). Three forms of biases frequently addressed in psychological and educational studies are acquiescence, careless responding, and social desirability bias (e.g., Cheung & Rensvold, 2000; Leite & Nazari, 2020; Wise, 2015). In formative assessments administered within virtual learning environments (VLE; Weller, 2007), it is hard to keep students motivated across different tests. Lack of motivation to complete formative assessments may result in careless responding, which in turn may produce biased estimates of student ability. This study focuses on careless responding, which occurs when students do not put much effort and thought into answering an assessment item (Voss & Vangsness, 2020). Wise and Kong (2005) have shown that careless responding can be associated with disengagement in an assessment. Other common names frequently used for this type of responding are non-effortful responding, inattentive responding, rapid guessing behavior, and examinee motivation (e.g., Rios & Soland, 2020; van Barneveld, 2007; Wise & DeMars, 2010; Wise, 2015; Wise and Kong, 2005). Throughout this paper, we use the "careless responding" label.

Researchers have proposed several different methods to identify careless responding. Attention check items including directed response items (e.g., "Please answer 'disagree' when responding to this item"), bogus items (e.g., "I do not understand a word of English," Meade & Craig, 2012), maximum longstring index, even-odd consistency, and Mahalanobis distance have been utilized to capture carelessness (e.g., Voss & Vangsness, 2020; Niessen et al. 2016). Self-report surveys have also been administered after the main test (e.g., Voss & Vangsness, 2020; Niessen et al. 2016; Wise, 2015), and response time effort and different types of time thresholds (Wise, 2017; Wise, 2015; Rios & Soland, 2020) are computed. Other researchers (e.g., Niessen et al., 2016; Patton et al., 2019) have used person-fit indices such as the number of Guttman errors and the standardized loglikelihood to detect careless responding. Since there is a wide range of available person-fit indices in the literature and each index captures misfitting persons from a different perspective, we opted for investigating person-fit indices.

Several nonparametric and parametric person-fit statistics (see Table 1) detect response biases that can be employed to detect carelessness (e.g., Karabatsos, 2003). Most of these indices were developed to target dichotomous items, and some of them have been extended to test polytomous items. Under the item response theory (IRT) framework, parametric person-fit indices assess the model fit at the individual level to examine the meaningfulness of obtained test scores (Embretson & Reise, 2013). In a way, the consistency of individuals' item response vectors is examined based on an IRT model by these indices (Embretson & Reise, 2013), which is the main concern when attempting to flag particular individual responses that may have been careless.

Although previous research on using person-fit indices to detect careless responding has provided promising results (Karabatsos, 2003), the current study attempted to improve the performance of person-fit indices by using multiple indices simultaneously. This is similar to considering multiple predictors in the model, which is possible by machine learning classifiers. Therefore, the present study sought to enhance capturing careless responding by using random forest (Breiman, 2001; Fernández-Delgado et al., 2014).). The study's research question is: Does the use of person-fit indices as features in a random forest to detect careless responding to items in formative assessments of a VLE outperform the use of person-fit indices by themselves? This

Sanaz Nazari, Walter Leite and Anne Huggins-Manley "Detecting Careless Responding to Assessment Items in a Virtual Learning Environment Using Person-fit Indices and Random Forest". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 635-640. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

research question was answered with an analysis of responses to multiple 10-item formative assessments within the Math Nation VLE (Lastinger Center for Learning & University of Florida, 2019).

Table 1: Used Person-fit indices in PerFit package

Person-fit index	Function	Author
Nonparametric		
Personal point-biserial correlation	r.pbis	Donlon & Fischer (1968)
Caution statistic	C.Sato	Sato (1975)
Modified caution statistic	Cstar	Harnisch & Linn (1981)
Number of Guttman errors	G	van der Flier, 1977
Normalized Guttman errors	Gnormed	van der Flier, 1977
Agreement statistic	A.KB	Kane & Brennan (1980)
Disagreement statistic	D.KB	Kane & Brennan (1980)
Dependability statistic	E.KB	Kane & Brennan (1980)
U3 statistic	U3	van der Flier (1980)
Standardized normal U3	ZU3	van der Flier (1982)
Norm conformity index	NCI	Tatsuoka & Tatsuoka (1982, 1983)
HT statistic	Ht	Sijtsma (1986), Sijtsma and Meijer (1992)
Parametric		
Standardized normal loglikelihood	lz	Drasgow et al. (1985)
Corrected lz	lzstar	Snijders (2001)

1.1 Theoretical Framework

A comparison of 36 person-fit indices was performed by Karabatsos (2003) to examine the strength of fit indices to detect five types of aberrant responding (cheating, careless responding, lucky guessing, creative responding, and random responding). Results showed that Ht (Sijtsma, 1986) and then U3 (Van Der Flier, 1982) person-fit indices produced the highest area under the curve (AUC) to detect all types of aberrant responding, and other indices such as Guttman errors (Meijer, 1994) and lz (Drasgow, Levine, & Williams, 1985) indicated acceptable AUC. Additionally, in a simulation study, Artner (2016) investigated five well-known indices to detect guessing, cheating, careless behavior, distorting, and fatigue in responses and found that Ht, Cstar, and U3 performed better than OUTFIT and INFIT. Recently, another comparative study (Beck, Albano & Smith, 2018) measured response time and mentioned person-fit statistics to detect inattentive responding. Again, Ht was found to have the highest AUC. However, a major limitation of using person-fit indices for detecting careless responding to formative assessment items is that they only classify the entire response vector for a quiz as careless or not. However, it may be that only part of the responses to an assessment was careless, such as the last few items due to respondent fatigue. To overcome this limitation, we propose the random forest approach that utilizes person-fit indices as predictors of careless responding.

Over the last decade, as educational datasets have become larger, alongside substantial increases in computation speed, researchers have shown interest in using more complex machine learning classifiers. The random forest was first introduced by Breiman (2001) to overcome the problems of boosting (i.e., fitting trees to bootstrap-resampled data, e.g., Shapire et al., 1998) and bagging (i.e., bootstrap aggregating: fitting trees to reweighted data; Breiman, 1996) by adding another layer of randomness to bagging. The advantage of the random forest is that it chooses the best predictor among a subset of randomly selected predictors for splitting a node, while in a standard tree, the best predictor is chosen among all variables (Liaw & Wiener, 2002). By using this strategy, the random forest is robust against overfitting, and it outperforms other classifiers, such as discriminant analysis, support vector machines, and neural networks in some situations (Breiman, 2001).

One major advantage of combining the random forest with person-fit indices to detect careless responding is that classification of responses as careless or not can be done at the item response level rather than the person-level. Therefore, for the vector of responses a student provides for a formative assessment, only some responses may be classified as careless. In other words, carelessness can be viewed less as a static person feature on a test and more as a feature of the interaction between a particular person and a particular assessment item. Another advantage of the proposed method is that it can simultaneously use multiple person-fit indices, allowing optimal use of each index's unique sensitivity to careless responding.

2. METHODS

2.1 Participants

The sample for this study consisted of item responses from 14474 students obtained from the Algebra 1 section of Math Nation during the time that face-to-face instruction in schools in the state of Florida was canceled due to the threat of COVID-19 infection and replaced by online instruction. More specifically, the period that responses were collected was from March 18 to May 31, 2020. Math Nation focuses on preparing students for taking the high-stakes Algebra 1 End-of-Course exam required for high school graduation, usually administered in May by the Florida Department of Education. Because of COVID-19, the Algebra 1 End-of-Course exam was canceled, removing an important motivator for students to use Math Nation. However, Math Nation usage spiked during this period because teachers could use it as a resource to teach algebra while students were attending classes virtually. Therefore, this sample is well-suited for studying careless responding because students were making heavy use of a VLE and its assessment features, and yet they did not have the pressure of practicing for the high-stakes achievement test at the end of the school year, and they were engaging in schooling from home, which may expose them to many distractions.

2.2 Measures

There are a total of 10 sections in Math Nation, which corresponds to major concepts in Algebra, such as linear functions and quadratic functions. Each section has between 6 and 12 formative assessments with 3 items and one formative assessment with 10 items randomly drawn uniquely for each student from an item pool. The students can take these assessments as many times as they like. This study's data included responses to 40 items of the item pool of the 10-item formative assessment of "Section 9: One-Variable Statistics" of Math Nation. We chose this section because it had the highest number of responses to items during the time period of interest. We chose to focus on the 10-item assessment rather than the 3-item ones because teachers frequently ask students to complete the 10-item assessment before moving forward to the next section of Math Nation. Once completed, students have the option to review their answers and watch solution videos for each item.

The items used in this study had multiple formats (e.g., single choice, multiple-choice, constructed response) but were scored dichotomously (i.e., correct, or incorrect). The items were written to mimic the content and format of items on the statewide Algebra 1 End-of-Course exam and have been found in our research project to correlate strongly to scores on that exam. Difficulty and discrimination parameters for the items in the study were taken from the estimates reported in a previous study (Xue et al., 2020), which used the 2-parameter logistic (2PL) item response theory model (IRT; Birnbaum, 1968).

2.3 Analysis

In the current study, we compared the effectiveness of 14 nonparametric and parametric person-fit statistics (see Table 1) as well as the performance of the random forest classifier to capture carelessly answered responses. These person-fit indices are available in the PerFit (Tendeiro, Meijer, & Niessen, 2016) package of R statistical software (R Core Team, 2018). To obtain nonparametric person-fit indices, no IRT model is required. However, for the parametric lz and lzstar, the item parameters (i.e., difficulty and discrimination) for a 2PL IRT model were obtained from Xue et al. (2020) and used to estimate the student ability parameters. The 2-PL model formula is

$$P(Y_{is} = 1|\theta_s) = \frac{exp^{1.7a_i(\theta_s - b_i)}}{1 + exp^{1.7a_i(\theta_s - b_i)}},$$

where P indicates probability, θ is a latent trait of ability, the subscript i indicates an item, Y is an item response, b is item difficulty, a is item discrimination, the subscript s indicates a student, and 1.7 is a scaling constant. This formula gives us the probability of correct response (Y=1) for item i and person s conditional on the individual's ability.

All 14 person-fit indices were computed using the *PerFit* (Tendeiro, Meijer, & Niessen, 2016) package for all selected individuals in the sample and the result of the person-fit indices (person-fit scores) was saved to be used as part of the predictors for the random forest. Additional predictors were item difficulty and discrimination estimated parameters, the number of items answered by each student, ranging from 10 to 40 items, and the number of correct items answered (from zero to 10). Within "Section 9: One-Variable Statistics", students could respond to all available 40 items by taking section tests multiple times. Only the responses for students who answered at least 10 items were retained for the analysis. The total number of answered items out of 40 were available in the dataset and used as a predictor of carelessness in the random forest.

Random forest was implemented with the *randomForest* (Liaw & Wiener, 2002) package in R. To identify carelessness in responses for the model training, the time taken by each student to answer each item was recorded, and by comparing the graph of the frequency of correct responses versus incorrect responses across time for each item, empirical cutoffs were determined. In most cases, the point in time where the frequency of incorrect responses was decreasing, and the frequency of correct responses started to increase, was chosen as a cutoff for carelessness. For example, for item two, a cutoff of eight seconds was determined (see blue dashed line in Figure 1). Therefore, student responses to item two that were recorded in less than eight seconds were coded as careless.

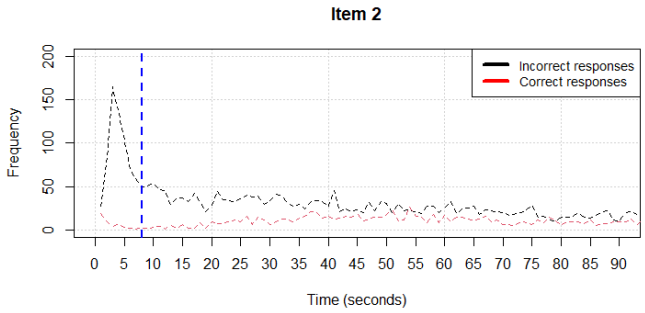


Figure 1: Frequency graph of correct responses versus incorrect responses across time for item 2.

To evaluate predictions by person-fit indices and random forest, we obtained the Receiver Operating Characteristic (ROC) curve. The ROC curve compares sensitivity against the specificity of a predictor for dichotomous data (Hanley, & McNeil, 1982). Sensitivity is the ability to identify true careless responding (true positive rate), and specificity is correctly identifying those items not careless (true negative rate). Usually, on ROC curves, "1-specificity" is demonstrated, which identifies the Type I Error rate (false positive rate; Hanley, & McNeil, 1982). The AUC of a ROC curve ranges from 0.5 to 1 from the identification line (i.e., the diagonal on the ROC) and represents the accuracy of the predictor or the feature, and different values of it can be interpreted to show the strength of a test. Generally, AUC that is 0.90-1 indicates an outstanding test, 0.80-0.90 is considered excellent, 0.70-0.80 is an acceptable one, and the AUC of about 0.5 suggests no discrimination (Hosmer, Lemeshow, & Sturdivant, 2013).

Creating ROC plots for person-fit indices requires the "true" careless responses (see above for flagging careless responses with time cutoffs) to compare them with estimated careless/non-careless responses by person-fit indices and calculate AUC. For person-fit indices, we defined a person as careless if he/she has responded to at least one item out of 10 carelessly based on the specified cutoffs. Regarding random forest ROC, true labels for carelessness were calculated per item response, and each student received 10 true labels of careless/non-careless for 10 answered items.

3. PRELIMINARY RESULT

Person-fit indices have been used as a measure to detect careless responding in previous studies (e.g., Niessen et al., 2016; Patton et al., 2019). In this study, all 14 available person-fit indices in the PerFit package were calculated for all students in the data, and the result are shown in Table 2. Most of these person-fit indices indicated a very poor AUC around 50%, which suggests no discrimination between careless/non-careless students. Therefore, only indices with the three best AUC were retained for later comparisons: 1) Guttman errors with 54.7%, 2) agreement statistic with 69.1%, and 3) dependability statistic with 63.5%. It is noteworthy that even the best performing index of agreement statistic with 69.1% AUC is not discriminating enough to be considered as an acceptable classifier of careless/non-careless students based on Hosmer, Lemeshow, and Sturdivant (2013) proposed cutoff of 70% AUC.

Table 2: AUC of 14 person-fit indices

Person-fit index	PerFit function	AUC (%)
Nonparametric		
Personal point-biserial correlation	r.pbis	50.2
Caution statistic	C.Sato	50.3
Modified caution statistic	Cstar	52.4
Number of Guttman errors	G	54.7
Normalized Guttman errors	Gnormed	49.9
Agreement statistic	A.KB	69.1
Disagreement statistic	D.KB	51.6
Dependability statistic	E.KB	63.5
U3 statistic	U3	52.4
Standardized normal U3	ZU3	50.1
Norm conformity index	NCI	50.1
HT statistic	Ht	50.4
Parametric		
Standardized normal loglikelihood	lz	50.1
Corrected lz	lzstar	50.1

The random forest has the advantage over person-fit indices in that it can use multiple person-fit indices as predictors but also include other predictors. The random forest included the three person-fit indices with the highest AUC and four additional predictors: estimated item difficulty and item discrimination parameters, number of correct responses, and number of items taken within the section. AUC of ROC illustrated that random forest with the set of specified predictors improved the classification and achieved the AUC of 77.2%, which is an acceptable test to distinguish between careless/non-careless responses (see Figure 2). The random forest also outperformed the best person-fit agreement statistic by about 8%.

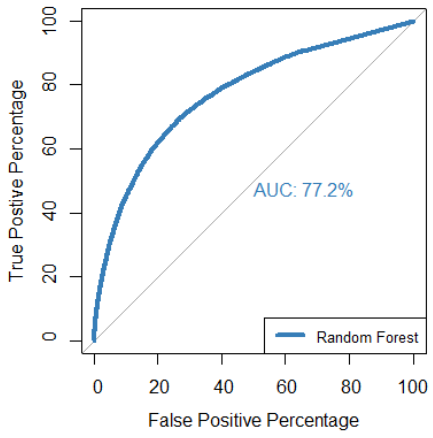


Figure 2: AUC of ROC plot for random forest classifier

4. DISCUSSION AND CONCLUSSION

The study objective was to compare the detection of careless responses between person-fit indices by themselves and random forest, including fit indices and other predictors. From the obtained results, we can conclude that random forest more accurately predicts careless responding, and this research took a methodological step forward in automatic identification of careless responding. By introducing different predictors, multiple dimensions are available, and a random forest can be used to investigate different parts of the data. In addition, careless responding can be conceptualized and examined as an item-person interaction rather than a static person feature. If desired, the random forest results can be aggregated to the person level (e.g., the proportion of responses from each person that were flagged as careless), allowing additional information at the person level. At this level, students can be labeled as careless responders or non-careless.

Person-fit indices as independent measures of carelessness may face some problems regarding ROC analysis. Sometimes, the true positive rates against false positive rates swing, cancel out, and end up in a very low AUC around 50% that is no discrimination. However, this issue does not occur in the random forest because of the way trees are constructed. Person-fit indices (and other predictors) can be removed from the random forest when they do not help improve classification accuracy.

One limitation of this study is that we relied on time cutoffs of each item response to create the criterion for careless responding. Like all options for "true" carelessness in responses, one could argue that our time-based flags of "true" careless responses are fallible in their own ways. However, empirical thresholds or time cutoffs have been used many times in previous research to detect careless responding (e.g., Wise & Kong, 2005; Wise & DeMars, 2010; Wise, 2015; Wise, 2017; Rios & Soland, 2020). Alternative criteria could come from surveys of students after they complete each formative assessment.

The result of this research will eventually be available as a trained random forest model to be used for applied researchers to detect careless responding in their data. They could enter the raw data to an R package to be developed in the future, and for the number of items in their test, they obtain a prediction of whether each answer is careless or non-careless. Then, within the context of their research, they can decide how they would like to aggregate and interpret carelessness (i.e., at the person level or at the item level) and make decisions according to the available results.

5. ACKNOWLEDGMENTS

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305C160004 to the University of Florida. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

6. REFERENCES

[1] Artner, R. (2016). A simulation study of person-fit in the Rasch model. *Psychological Test and Assessment Modeling*, *58*(3), 531-563. Retrieved from https://lirias.kuleuven.be/retrieve/523896

[2] Beck, M. F., Albano, A. D., & Smith, W. M. (2019). Person-Fit as an Index of Inattentive Responding: A Comparison of Methods Using Polytomous Survey Data. *Applied Psychological Measurement*, *43*(5), 374-387.. https://doi.org/10.1177%2F0146621618798666

[3] Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability, Contributed

chapter. *Statistical theories of mental test scores*, Chapters-17.

[4] Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140. https://doi.org/10.1007/BF00058655

[5] Breiman, L. Random Forests. *Machine Learning* **45,** 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[6] Cheung, G. W., & Rensvold, R. B. (2000). Assessing extreme and acquiescence response sets in cross-cultural research using structural equations modeling. *Journal of cross-cultural psychology*, *31*(2), 187-212. https://doi.org/10.1177%2F0022022100031002003

[7] Donlon, T. F., & Fischer, F. E. (1968). An index of an individual's agreement with group-determined item difficulties. *Educational and Psychological Measurement*, *28*(1), 105-113. https://psycnet.apa.org/doi/10.1177/001316446802800110

[8] Drasgow, F., Levine, M. V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology*, *38*(1), 67-86. https://doi.org/10.1111/j.2044-8317.1985.tb00817.x

[9] Embretson, S. E., & Reise, S. P. (2013). *Item response theory*. Psychology Press.

[10] Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?. *The journal of machine learning research*, *15*(1), 3133-3181. https://doi.org/10.1117/1.JRS.11.015020

[11] Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*(1), 29-36. https://doi.org/10.1148/radiology.143.1.7063747

[12] Harnisch, D., & Linn, R. (1981). Analysis of Item Response Patterns: Questionable Test Data and Dissimilar Curriculum Practices. *Journal of Educational Measurement, 18*(3), 133-146. Retrieved October 15, 2020, from http://www.jstor.org/stable/1434737

[13] Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (Vol. 398). John Wiley & Sons.

[14] Kane, M. T., & Brennan, R. L. (1980). Agreement coefficients as indices of dependability for domain-referenced tests. *Applied Psychological Measurement*, *4*(1), 105-126. https://doi.org/10.1177%2F014662168000400111

[15] Karabatsos, G. (2003). Comparing the aberrant response detection performance of thirty-six person-fit statistics. *Applied Measurement in Education*, *16*(4), 277-298. https://doi.org/10.1207/S15324818AME1604_2

[16] Lastinger Center for Learning, & University of Florida. (2019). *Algebra Nation*. Retrieved 9/20/2019 from http://lastingercenter.com/portfolio/algebra-nation-2/

[17] Leite W.L., Nazari S. (2020) Marlowe-Crowne Social Desirability Scale. In: Zeigler-Hill V., Shackelford T.K. (eds) Encyclopedia of Personality and Individual Differences. Springer, Cham. https://doi.org/10.1007/978-3-319-24612-3_45

[18] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news*, *2*(3), 18-22. Retrieved from https://www.r-project.org/doc/Rnews/Rnews_2002-3.pdf

[19] Meade, A. W., & Craig, S. B. (2012). Identifying careless responses in survey data. *Psychological methods*, *17*(3), 437. http://dx.doi.org/10.1037/a0028085

[20] Meijer, R. R. (1994). The number of Guttman errors as a simple and powerful person-fit statistic. *Applied Psychological Measurement*, *18*(4), 311-314. https://doi.org/10.1177%2F014662169401800402

[21] Patton, J. M., Cheng, Y., Hong, M., & Diao, Q. (2019). Detection and treatment of careless responses to improve item parameter estimation. *Journal of Educational and Behavioral Statistics*, *44*(3), 309-341. https://doi.org/10.3102%2F1076998618825116

[22] Paulhus, D. L. (1991). Measures of personality and social psychological attitudes. In J. P. Robinson & R. P. Shaver (Eds.), *Measures of social psychological attitudes series* (Vol. 1, pp. 17–59). San Diego: Academic. https://doi.org/10.1016/C2013-0-07551-2

[23] R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Available online at https://www.R-project.org/.

[24] Rios, J. A., & Soland, J. (2020). Parameter Estimation Accuracy of the Effort-Moderated Item Response Theory Model Under Multiple Assumption Violations. *Educational and Psychological Measurement*, 0013164420949896. https://doi.org/10.1177%2F0013164420949896

[25] Sato, T. (1975). The construction and interpretation of SP tables. Tokyo, Japan: Meiji Tosho.

[26] Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, *26*(5), 1651-1686. https://doi.org/10.1214/aos/1024691352

[27] Sijtsma, K. (1986). A coefficient of deviance of response patterns. *Kwantitatieve Methoden*, *7*(22), 131-145. Retrieved from https://research.tilburguniversity.edu/files/1030745/COEFFICI.PDF

[28] Sijtsma, K., & Meijer, R. R. (1992). A method for investigating the intersection of item response functions in Mokken's nonparametric IRT model. *Applied Psychological Measurement*, *16*(2), 149-157. https://doi.org/10.1177%2F014662169201600204

[29] Snijders, T. A. (2001). Asymptotic null distribution of person fit statistics with estimated person parameter. *Psychometrika*, *66*(3), 331-342. https://doi.org/10.1007/BF02294437

[30] Tatsuoka, K. K., & Tatsuoka, M. M. (1982). Detection of aberrant response patterns and their effect on dimensionality. *Journal of Educational Statistics*, *7*(3), 215-231. https://doi.org/10.3102%2F10769986007003215

[31] Tatsuoka, K. K., & Tatsuoka, M. M. (1983). Spotting erroneous rules of operation by the individual consistency index. *Journal of Educational Measurement*, 221-230.

[32] Tatsuoka, K., & Tatsuoka, M. (1983). Spotting Erroneous Rules of Operation by the Individual Consistency Index. *Journal of Educational Measurement, 20*(3), 221-230.

Retrieved October 15, 2020, from http://www.jstor.org/stable/1434713

[33] Tendeiro, J. N., Meijer, R. R., & Niessen, A. S. M. (2016). PerFit: An R package for person-fit analysis in IRT. *Journal of Statistical Software*, *74*(5), 1-27. http://dx.doi.org/10.18637/jss.v074.i05

[34] van Barneveld, C. (2007). The effect of examinee motivation on test construction within an IRT framework. *Applied Psychological Measurement*, *31*(1), 31-46. https://doi.org/10.1177%2F0146621606286206

[35] Van der Flier, H. (1977). Environmental factors and deviant response patterns. *Basic problems in cross cultural psychology, Amsterdam: Swets & Seitlinger*.

[36] Van der Flier, H. (1980). *Vergelijkbaarheid van individuele testprestaties*. Swets & Zeitlinger.

[37] Van der Flier, H. (1982). Deviant response patterns and comparability of test scores. *Journal of Cross-Cultural Psychology*, *13*(3), 267-298. https://doi.org/10.1177%2F0022002182013003001

[38] Voss, N. M., & Vangsness, L. (2020). Is Procrastination Related to Low-Quality Data?. *Educational Measurement: Issues and Practice*. https://doi.org/10.1111/emip.12355

[39] Weller, M. (2007). *Virtual learning environments: Using, choosing and developing your VLE*. Routledge.

[40] Wise, S. L. (2015). Effort analysis: Individual score validation of achievement test data. *Applied Measurement in Education*, *28*(3), 237-252. https://doi.org/10.1080/08957347.2015.1042155

[41] Wise, S. L. (2017). Rapid-guessing behavior: Its identification, interpretation, and implications. *Educational Measurement: Issues and Practice*, *36*(4), 52-61. https://doi.org/10.1111/emip.12165

[42] Wise, S. L., & DeMars, C. E. (2010). Examinee noneffort and the validity of program assessment results. *Educational Assessment*, *15*(1), 27-41. https://doi.org/10.1080/10627191003673216

[43] Wise, S. L., & Kong, X. (2005). Response time effort: A new measure of examinee motivation in computer-based tests. *Applied Measurement in Education*, *18*(2), 163-183. https://doi.org/10.1207/s15324818ame1802_2

[44] Xue, K., Huggins-Manley, A. C., & Leite, W. L. (2020). Semi-supervised Learning Method for Adjusting Biased Item Difficulty Estimates Caused by Nonignorable Missingness under 2PL-IRT Model In A. N. Rafferty, J. Whitehill, C. Romero, & V. Cavalli-Sforza (Eds.), *Proceedings of The 13th Conference of Educational Data Mining*. https://educationaldatamining.org/files/conferences/EDM2020/papers/paper_217.pdf

Tracing Knowledge for Tracing Dropouts: Multi-Task Training for Study Session Dropout Prediction

Seewoo Lee*
Riiid! AI Research
seewoo.lee@riiid.co

Kyu Seok Kim*
Riiid! AI Research
kyuseok.kim@riiid.co

Jamin Shin
Riiid! AI Research
jamin.shin@riiid.co

Juneyoung Park
Riiid! AI Research
juneyoung.park@riiid.co

ABSTRACT

Study session dropout prediction allows for educational systems to identify when a student would stop a study session which gives vital information to prolong learning activity. Student session dropout can depend on many factors that are involved with the engagement when using the system. The student's knowledge level and their track records within the system are closely related to the student's willingness to continue with their study. Knowledge tracing as a task models the user's knowledge level given study history. The information from knowledge tracing can have significant impact on predicting the student's willingness to continue, which is why it is natural to train two tasks jointly for better generalization in dropout prediction task. While extensive research has been conducted individually on dropout prediction and knowledge tracing, the effect of jointly modeling two tasks has not been thoroughly investigated. Hence, we show that multi-task training of the study session dropout prediction model along with knowledge tracing boosts the performance of study session dropout prediction, especially on more challenging tasks and datasets. Specifically, with Transformer-based models, multi-task training significantly improves Area Under Receiving Operator Curve (AUROC) by 3.62% in further N-step dropout prediction task, which is a study session dropout prediction task under a more practical setting. Moreover, under label-scarce and class-imbalance settings, our method shows improvements of AUROC up to 12.41% and 11.22%, respectively. Our results imply that knowledge tracing is closely related to study session dropout prediction and can transfer positive knowledge in multi-task training, which provides a new way to better predict dropouts especially in difficult settings.

Keywords

Dropout prediction, Multi-task Training, Knowledge Tracing

1. INTRODUCTION

The advantages of e-learning has gathered the attention of both educators and researchers. One of the lasting problems in e-learning is the ability to maintain the user's attention during the system use. For instance, students in mobile learning environments are more prone to distractions and exhibit difficulties in concentration [16, 20, 5]. Thus, being able to properly identify when these issues occur will allow an Intelligent Tutoring System (ITS) [1] to appropriately and preemptively intervene. This task is called Study Session Dropout Prediction and has been recently proposed by [21]. Predicting such session dropout is a crucial task in Educational Data Mining (EDM) to understand student's behaviors and learning environments, which can lead to increased learning effect.

However, study session dropout prediction has not yet been extensively studied. Many recent research works have instead focused on predicting student dropout in environments like universities or Massive Open Online Courses (MOOC) [3, 13, 27, 33, 35]. Interestingly, [22, 9, 26] has also shown that one of the main reasons students drop out from schools or classes is their academic performance which is highly relevant to their knowledge states. Given such knowledge, we hypothesize that study session dropout can also be attributed to the knowledge states of students.

Hence, in this paper, we jointly model study session dropout prediction with knowledge tracing, which is a heavily studied task that [12, 23, 8] that aims to predict the student's future performance on *knowledge components* (e.g. questions or concepts) given the student's historical data. In this study, we address this issue through a machine learning methodology known as multi-task learning [4]. Multi-task learning jointly trains multiple tasks together to formulate a comprehensive understanding of the nature of the data. Specifically, we implement a multi-task training model that is trained with both session dropout prediction and knowledge tracing.

The contributions of this paper are as such:

- We provide a multi-task training framework to jointly model study session dropout prediction and knowledge tracing.
- We show that our multi-task training framework boosts the performance of the trained model on study session dropout prediction. Also, we show that our method elevates the performance in further N-step dropout prediction task, where the model has to predict dropouts not only in immediate time step, but also in future time steps.
- We perform extensive ablation studies to show that multi-task training shows even higher performance on more difficult experimental settings such as label scarcity and class-imbalance. We also show with ablation studies that the thresh-

Seewoo Lee, Kyu Seok Kim, Jamin Shin and Juneyoung Park "Tracing Knowledge for Tracing Dropouts: Multi-Task Training for Study Session Dropout Prediction". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 641-647. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

old of lag time that we use to define session dropout also affects the performance of multi-task training.

2. RELATED WORKS

2.1 Dropout prediction

There have been many attempts to predict dropouts in various environments. Traditionally, dropout prediction has been incorporated to predict student dropouts [3, 27]. Following the proliferation of internet, dropout prediction became applicable in online services. [2, 15] incorporated deep learning methods in *Spotify Sequential Skip Prediction Challenge*, where the task is to infer the songs that will be skipped in the second half session after the first half. Models such as denoising autoencoder and variants of Long Short Term Memory (LSTM) network were utilized. [13, 33, 35] used deep learning methods such as Convolutional Neural Network (CNN) to predict students' dropouts in MOOC. Study session dropout prediction has been studied to discover student's involvements in mobile learning environments. [21] utilized the Transformer network [31], which replaced the recurrent architecture of Recurrent Neural Networks (RNN) by self-attention blocks, to predict the study session dropout probability in mobile learning environment. We used **D**eep **A**ttentive Study **S**ession Dropout prediction (DAS) model in [21] with multi-task training approach in our experiments.

2.2 Multi-Task training

Multi-task training or multi-objective training is a method to train a machine learning model with multiple objectives [37], which tries to enhance the performance on original task by sharing features with auxiliary tasks. It has been used across various domains in machine learning, such as Computer Vision and Natural Language Processing. For example, in [24], the authors used multi-task training for face labeling by training a CNN to handle both likelihoods and pairwise label dependencies. Multi-Task Deep Neural Network (MT-DNN, [25]), is a BERT [11] based model with several task-specific layers for multi-task learning, which outperforms vanilla BERT's performance on the GLUE benchmark [32].

There are also several applications of multi-task training in an educational field. Huang et al. [18] presents a transformer-based model that identifies whether a given voice of a teacher corresponds to a question or not, which solve the problem as a multi-class classification problem to recognize question types. Geden et al. [14] proposed LSTM-based model to predict correctness rate of all questions instead of the average correctness rate for the related questions. In [19], Huang et al. suggests Deep Reinforcement Learning based exercise recommendation system whose reward function is designed to satisfy multiple objectives.

2.3 Knowledge Tracing

Knowledge tracing is a task of modeling students' knowledge level given their learning activities. Knowledge tracing and dropout prediction shares the aspect that they both model students' responses given their learning histories. Bayesian Knowledge Tracing (BKT) is a traditional method which treats student's learning activities as binary variables representing whether the student understands a certain concept or not [36]. Some works proposed to incorporate deep learning methods in knowledge tracing. [29] feeds the users' one-hot encoded learning activities into RNN-based model architectures to output the correctness prediction probability. [7, 28] are the works that use Transformer-based architectures for knowledge tracing. SAINT [7] has a similar architecture to DAS, which uses Transformer's both encoder and decoder structure.

3. METHODS

3.1 Study Session Dropout Prediction

Formally, a student's learning history is given as a sequence of interactions

$$I = (I^{(1)}, I^{(2)}, I^{(3)}, \ldots, I^{(T)})$$

where each $I^{(j)} = (e^{(j)}, l^{(j)})$ includes meta-data of the question $e^{(j)}$ that a student solves at j-th step (e.g. question id, category of the question, question text, ...) and the meta-data of the student's response $l^{(j)}$ (e.g. response correctness, elapsed time, timeliness, ...) at j-th step. Then the *study session dropout prediction* is to estimate the probability

$$\mathbb{P}[y_{DP}^{(j)} = 1 | I^{(1)}, I^{(2)}, \ldots, I^{(j-1)}, e^{(j)}]$$

that the session dropout occurs after solving j-th question. Note that a sequence can contain multiple sessions. As in [21], we define one-hour inactivity as a session dropout, so that the dropout label at j-th step is given by

$$y_{DP}^{(j)} = \begin{cases} 1 & lt^{(j)} := st^{(j+1)} - st^{(j)} \geq 1 \text{ hour} \\ 0 & \text{otherwise} \end{cases}$$

where $st^{(j)}$ is the *start time* at j-th step, i.e. the time that user start to solve the question, and $lt^{(j)}$ is the lag time for the j-th interaction.

3.2 Input Representation

The representation of each interaction $I^{(j)} = (e^{(j)}, l^{(j)})$ is formulated similarly to the settings in [21]. Here are some minor differences of feature settings between our model and the original DAS model in [21]:

1. Instead of *start time*, we use the *lag time* feature. It is more directly related to the dropout and leads to the substantial gain in the model's performance. Since the distribution of a lag time is long-tailed, we use the logarithm of the lag time instead of the lag time itself. (See Figure 2 for the distribution of the lag time). It is used as a decoder's input, not for an encoder.
2. We use *continuous embedding* for *elapsed time*, instead of discrete embedding. More precisely, we first clip the actual elapsed time with maximum 300 seconds, then normalize it by dividing it with 300. After that, we get a latent embedding vector for the elapsed time et by $\mathbf{v} = \mathbf{v}(\mathrm{et}) = \mathrm{et} \cdot \mathbf{w}_{et}$, where $\mathbf{w}_{et}$ is a single trainable vector which has same dimension as the model.

3.3 Model

In this section, we describe our methodology to jointly perform training in dropout prediction and knowledge tracing. We use the shared model f to generate the shared feature representations for both dropout prediction and knowledge tracing. An arbitrary model f takes the sequences of question embeddings $e = [e^{(1)}, \ldots, e^{(j)}]$ and response embeddings $l = [l^{(1)}, \ldots, l^{(j-1)}]$ to produce the shared feature representation for dropout prediction and knowledge tracing. Then, the feature representation is fed into the final separate prediction layers to output predicted dropout probabilities and response correctness:

$$\hat{y}_{DP} = \sigma\big(\mathbf{W}_{\mathrm{DP}}(f(e,l)) + \mathbf{b}_{DP}\big)$$
$$\hat{y}_{KT} = \sigma\big(\mathbf{W}_{\mathrm{KT}}(f(e,l)) + \mathbf{b}_{KT}\big)$$

Further Steps	$N = 1$ (Positive label 4.06%)				$N = 5$ (Positive label 18.34%)				$N = 10$ (Positive label 32.52%)			
model	AUROC		AUPRC		AUROC		AUPRC		AUROC		AUPRC	
	vanilla	multi-objective	vanilla	multi-objective	vanilla	multi-objective	vanilla	multi-objective	vanilla	multi-objective	vanilla	multi-objective
LSTM	0.8704	**0.8717**	0.3208	**0.3229**	0.7586	**0.7599**	0.4577	**0.4598**	0.736	**0.7367**	0.5880	**0.5886**
GRU	0.8652	**0.8670**	0.3083	**0.3133**	0.7567	**0.7570**	0.4547	**0.4556**	0.7338	**0.7349**	0.5842	**0.5855**
DAS	0.8807	**0.8836**	0.3469	**0.3534**	0.7579	**0.7734**	0.4598	**0.4815**	0.7229	**0.7491**	0.5717	**0.6061**

Table 1: Test AUROCs and AUPRCs of DAS and RNN-based dropout prediction models. Further steps N of each task and its positive label proportions of the dataset are indicated in the top row ($N = 1$ corresponds to the original session dropout prediction task). Best result for each model is indicated in bold.

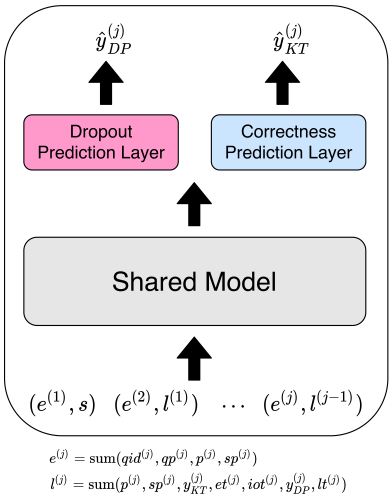


Figure 1: Overall architecture of our multi-task training scheme. Note that s is the starting token.

The major difference between our method and the previous dropout prediction models is that we are jointly training the model to predict both student dropout *and* response correctness, by using different prediction layer for each task at the end of the shared model. Using separate prediction layers, we produce both $\hat{y}_{DP} = [\hat{y}_{DP}^{(1)}, \ldots, \hat{y}_{DP}^{(j)}]$ and $\hat{y}_{KT} = [\hat{y}_{KT}^{(1)}, \ldots, \hat{y}_{KT}^{(j)}]$, which are predicted probabilities for study session dropout ($\hat{y}_{DP}$) and response correctness ($\hat{y}_{KT}$) for each time step. Training scheme of our approach is described in Figure 1. The major baseline that we use for our methodology is DAS [21], which is a Transformer-based model to predict study session dropout. The details of the architecture of DAS is described in Appendix A. We also do experiments with RNN-based model architectures - including LSTM and GRU [6, 17] - which are provided as baselines in [21] for comparison. For RNN-based models, we use encoder-only structure instead of encoder-decoder structure.

3.4 Training objectives

Typically, Binary Cross-Entropy (BCE) loss is used in 2-class classification tasks, which include the cases of dropout prediction and knowledge tracing. We use the BCE function to compute $\mathcal{L}_{\text{DP}}$ and $\mathcal{L}_{\text{KT}}$, which are the losses for dropout prediction and knowledge tracing. We train the model with the loss

$$\mathcal{L} = \mathcal{L}_{\text{DP}} + \lambda_{\text{KT}} \mathcal{L}_{\text{KT}}$$

where λ_{KT} is a balancing hyper-parameter. Our experiments are performed with $\lambda_{\text{KT}} = 0.5$.

4. EXPERIMENTS

4.1 Experiment setup

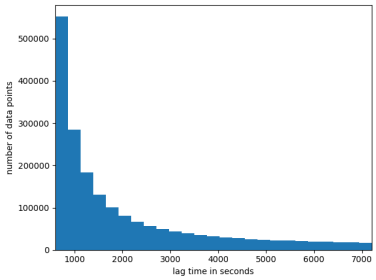


Figure 2: Distribution of data points with respect to the lag time. The lag time in the graph ranges from 600s (10 minutes) to 7200s (2 hours). The number of data points exponentially decays as their lag time increases.

mask rate	Positive Label Proportion	AUROC		AUPRC	
		vanilla	multi-objective	vanilla	multi-objective
50%	3.80%	0.8381	**0.8832**	0.2599	**0.3521**
90%	3.80%	0.7752	**0.8740**	0.1602	**0.3304**
95%	3.80%	0.7264	**0.7598**	0.1202	**0.1322**
99%	3.81%	0.6826	**0.6837**	**0.0984**	0.0901
50%	1.90%	0.8418	**0.8833**	0.2775	**0.3520**
90%	0.38%	0.7677	**0.8204**	0.1504	**0.2093**
95%	0.19%	0.7175	**0.7980**	0.1163	**0.1726**
99%	0.04%	0.6782	**0.7304**	0.0923	**0.1095**

Table 2: Test AUROCs and AUPRCs of the DAS model with various masking rates on *both labels (first 4 rows)* and *only positive dropout labels (last 4 rows)*. The first two columns indicate the rate of random masking and the proportion of labels in the training data for each mask rate. Best result for each masking rate indicated in bold.

We use the EdNet-KT1 dataset [8], the largest publicly available student interaction dataset collected by *Santa*[*], which is a mobile application for preparing Test of English for International Communication (TOEIC) exam. The proportion of the logs where session dropout occurred was 4.06% with the definition of dropout as one-hour lag time. The distribution of dropout labels w.r.t. the change of lag time in defining the dropout is described in Figure 2.

For RNN-based model architectures, we use the embedding and model dimension size of 256 and feedforward layer dimension size of 1024 with 2 number of layers. For DAS, we use the embedding and model dimension size of 512 and feedforward layer dimension size of 2048 with 4 number of layers. While training, we set the model's input sequence size as 100, and all the models are trained with Adam optimizer with Noam scheduling where the warmup step is 40000. We set the initial learning rate and the model's dropout rate as 0.001 and 0.1 respectively.

We evaluate our models with two metrics: Area Under Receiving Operator Curve (AUROC) and Area Under Precision Recall Curve (AUPRC). AUROC is the most widely used metric in the litera-

[*]https://aitutorsanta.com/

lag time	Positive Label Proportion	AUROC		AUPRC	
		vanilla	multi-objective	vanilla	multi-objective
600s	6.33%	0.8853	**0.8876**	0.4327	**0.4387**
1800s	4.61%	0.8786	**0.8821**	0.3582	**0.3657**
3600s	4.06%	0.8807	**0.8836**	0.3469	**0.3534**
5400s	3.79%	0.8768	**0.8857**	0.3309	**0.3519**
7200s	3.60%	0.8789	**0.8876**	0.3298	**0.3508**

Table 3: Test AUROCs and AUPRCs of the DAS model with various standards of lag time on defining dropouts (in seconds). The first two columns indicate the lag time used to define session dropout and the proportion of positive labels in the total dataset for each definition. Best result for each indicated in bold.

ture for evaluating the dropout prediction models because labels in dropout prediction settings are usually imbalanced. It is known that AUPRC is especially more informative than the AUROC when the dataset's labels are imbalanced [10, 30].

We evaluate the effect of multi-task training on two tasks. The first is the standard study session dropout prediction described in 3.1, where the task is to estimate the probability that the session dropout occurs after the current time step. However, in the actual service, it is more important to predict whether the user will dropout within several future time steps in order to respond to the user's engagement in advance. Thus, we also evaluate our method on *further N-step dropout prediction* task, which predicts whether the user will dropout within further N time steps. In further N-step dropout prediction, the number of future time steps that the model has to consider increases as N increases. We perform further N-step dropout predictions with $N \in \{5, 10\}$. For all tasks, we measure AUROC and AUPRC on LSTM, GRU, and DAS with and without multi-task training to validate the effect of our method.

4.2 Main results

The results of multi-task training on the study session dropout prediction are given in Table 1. The multi-task training with knowledge tracing improves AUROC and AUPRC for the dropout prediction across all models. We also present the effect of our method in further N-step dropout predictions in Table 1. The results show that in further N-step dropout prediction tasks, multi-task training increases the performance of the model by larger margins than in immediate dropout prediction task. Note that multi-task training shows higher increase in AUROC when $N = 10$ since the future steps that the model has to consider increases with N, leaving more room for multi-task training to help the model. Table 1 also includes the proportion of positive labels of the dataset in each task to explain the difference of AUPRCs between the tasks. Although further 10-step prediction shows lower AUROCs compared to other tasks, since its dataset is less imbalanced, it shows higher AUPRCs.

4.3 Ablation study

We performed ablation studies on immediate study session dropout prediction task for fair comparisons. We perform ablations on label scarcity, imbalanced datasets, and various standards on dropout definition as follows.

4.3.1 Scarce Label for Dropout Prediction

It has been known that multi-task training shows higher performance when the label of target domain is scarce [34]. To verify this notion, we evaluated the multi-task training on datasets with different levels of dropout prediction label scarcity. Specifically, we randomly masked out both positive and negative dropout labels in different proportions ranging in {50%,90%,95%,99%}. The results in Table 2 shows that multi-task training indeed shows higher performance when dropout prediction labels are scarce. Since at least some amount of labels are needed for the models to converge, the result with 99% mask rate fails to show meaningful results.

4.3.2 Imbalanced Dataset

As we mentioned before, study session dropout prediction usually suffers from the imbalanced dataset. In our case, the rate of the positive label is only 4.06% of the total data. We conjecture that our multi-task training approach is also helpful when the label of the dataset is extremely imbalanced. To show this, while training, we randomly masked out certain proportion of **positive** dropout labels during training, and evaluated the model on the same validation and test set as before. Note that this is different from 4.3.1 since 4.3.1 performs random masking on both positive and negative dropout prediction labels. The proportion of random masking also ranges in {50%,90%,95%,99%}. The results are given in the Table 2. Results show that multi-task training outperforms vanilla model more heavily on imbalanced datasets.

4.3.3 Definition on Dropout

Although we define one-hour (3600s) inactive lag time as a session dropout, other definitions of a dropout may be utilized to better analyze student's learning activities. Thus, we see how the effect of multi-task training varies with the change in the definition of a session dropout. Figure 2 shows the distribution of the number of dropout labels w.r.t. the inactivity duration (lag time). We compare the results with various lag time standards of a session dropout in $\{600s, 1800s, 3600s, 5400s, 7200s\}$. The results are given in Table 3. Results show that multi-task training tends to perform better in tasks with higher inactivity duration standards of a session dropout. This is because the tasks with higher lag time standards have more imbalanced datasets. Since imbalanced datasets tend to have lower AUPRC, tasks with higher lag time standards have lower AUPRCs.

5. CONCLUSIONS

In this paper, we proposed a multi-task training approach with knowledge tracing to boost the performance of study session dropout prediction. We hypothesized that the commonality between the dropout prediction and knowledge tracing tasks would be beneficial to predict dropouts. We empirically validated with Transformer-based and RNN-based models that multi-task training enhances the dropout prediction performance especially in further N-step dropout prediction, which is a more practical task in real service. Moreover, we performed extensive ablation studies to demonstrate that multi-task training shows even better performance on more difficult experimental settings. We remain the multi-task training with other tasks in the field of Artificial Intelligence in Education (AIEd) as the future work.

6. REFERENCES

[1] J. R. Anderson, C. F. Boyle, and B. J. Reiser. Intelligent tutoring systems. *Science*, 228(4698):456–462, 1985.

[2] F. Beres, D. M. Kelen, and A. A. Benczur. Sequential skip prediction using deep learning and ensembles. In *International Conference on Web Search and Data Mining*, 2019.

[3] F. D. Bonifro, M. Gabbrielli, G. Lisanti, and Z. Stefano. Student dropout prediction. In *International Conference on*

Artificial Intelligence in Education, pages 129–140. Springer, 2020.

[4] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[5] Q. Chen and Z. Yan. Does multitasking with mobile phones affect learning? a review. *Computers in Human Behavior*, 54:34–42, 2016.

[6] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[7] Y. Choi, Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and H. Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the Seventh ACM Conference on Learning*, pages 341–344, 2020.

[8] Y. Choi, Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, C. Bae, B. Kim, and J. Heo. Ednet: A large-scale hierarchical dataset in education. In *International Conference on Artificial Intelligence in Education*, pages 69–73. Springer, 2020.

[9] F. J. da Costa, M. de Souza Bispo, and R. de Cássia de Faria Pereira. Dropout and retention of undergraduate students in management: a study at a brazilian federal university. *RAUSP Management Journal*, 53(1):74–85, 2018.

[10] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning.*, pages 233–240, 2006.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[12] M. Feng, N. Heffernan, and K. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User modeling and user-adapted interaction*, 19(3):243–266, 2009.

[13] W. Feng, J. Tang, and T. X. Liu. Understanding dropouts in moocs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 517–524, 2019.

[14] M. Geden, A. Emerson, J. Rowe, R. Azevedo, and J. Lester. Predictive student modeling in educational games with multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 654–661, 2020.

[15] C. Hansen, C. Hansen, S. Alstrup, J. G. Simonsen, and C. Lioma. Modelling sequential music track skips using a multi-rnn approach. In *International Conference on Web Search and Data Mining*, 2019.

[16] B. A. Harman and T. Sato. Cell phone use and grade point average among undergraduate university students. *College Student Journal*, 45(3):544–550, 2011.

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[18] G. Y. Huang, J. Chen, H. Liu, W. Fu, W. Ding, J. Tang, S. Yang, G. Li, and Z. Liu. Neural multi-task learning for teacher question detection in online classrooms. In *International Conference on Artificial Intelligence in Education*, pages 269–281. Springer, 2020.

[19] Z. Huang, Q. Liu, C. Zhai, Y. Yin, E. Chen, W. Gao, and G. Hu. Exploring multi-objective exercise recommendations in online education systems. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1261–1270, 2019.

[20] R. Junco. Too much face and not enough books: The relationship between multiple indices of facebook use and academic performance. *Computers in human behavior*, 28(1):187–198, 2012.

[21] Y. Lee, D. Shin, H. Loh, J. Lee, P. Chae, J. Cho, S. Park, J. Lee, J. Baek, B. Kim, et al. Deep attentive study session dropout prediction in mobile learning environment. *arXiv preprint arXiv:2002.11624*, 2020.

[22] S. A. Lim and R. Rumberger. Why students drop out of school: A review of 25 years of research. 2008.

[23] R. V. Lindsey, M. Khajah, and M. C. Mozer. Automatic discovery of cognitive skills to improve the prediction of student learning. In *Advances in neural information processing systems*, pages 1386–1394. Citeseer, 2014.

[24] S. Liu, J. Yang, C. Huang, and M.-H. Yang. Multi-objective convolutional learning for face labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3451–3459, 2015.

[25] X. Liu, P. He, W. Chen, and J. Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, 2019.

[26] M. Manacorda. Grade failure, drop out and subsequent school outcomes: quasi-experimental evidence from uruguayan administrative data. 2006.

[27] C. Márquez-Vera, A. Cano, C. Romero, A. Y. M. Noaman, H. Mousa Fardoun, and S. Ventura. Early dropout prediction using data mining: a case study with high school students. *Expert Systems*, 33(1):107–124, 2016.

[28] S. Pandey and G. Karypis. A self-attentive model for knowledge tracing. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 384–389, 2019.

[29] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, volume 01, pages 505–513, 2015.

[30] T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and P. Illia. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.

[32] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[33] W. Wang, H. Yu, and C. Miao. Deep model for dropout prediction in moocs. In *Proceedings of the 2nd International Conference on Crowd Science and Engineering*, pages 26–32, 2017.

[34] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell. Characterizing and avoiding negative trasfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern*

Recognition, 2019.

[35] J. Whitehill, K. Mohan, D. Seaton, Y. Rosen, and D. Tingley. Mooc dropout prediction: How to measure accuracy? In *Proceedings of the Fourth ACM Conference on Learning*, pages 161–164, 2017.

[36] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *International Conference on Artificial Intelligence in Education*, pages 171–180. Springer, 2013.

[37] Y. Zhang and Q. Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

APPENDIX

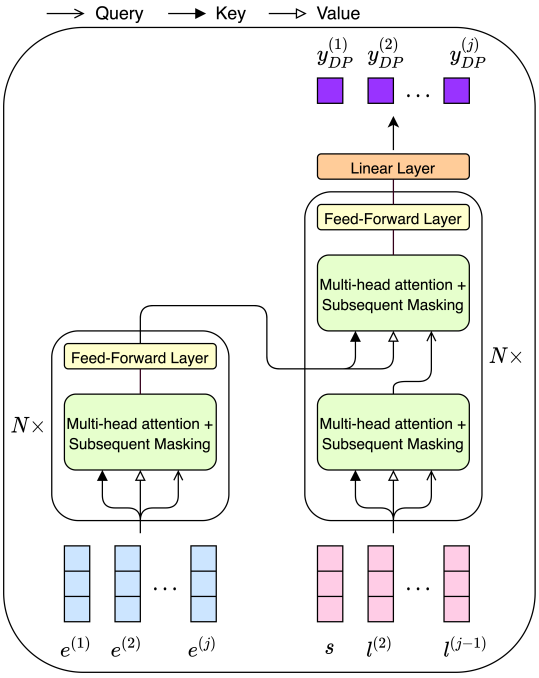


Figure 3: Overall architecture of DAS. Note that s is the starting token and N is the number of layers.

A. ARCHITECTURE OF DAS

In this section, we review the overall architecture of Deep Attentive Study Session Prediction (DAS) [21], which we use for the baseline of our model. DAS is a transformer-based model that consists of an encoder and a decoder. Encoder includes N encoder blocks where each block has a multi-head self-attention layer and a fully connected feed-forward layer. After each layer, residual connection and layer normalization are applied. Decoder also includes N decoder blocks with each block including a multi-head self-attention layer, a multi-head encoder-decoder attention layer, and a fully connected feed-forward layer. The encoder-decoder attention layer takes the output of the encoder as keys and values, and output of self-attention layer as queries to perform the attention mechanism. Each layer in the decoder block is also followed by residual connection and layer normalization. The encoder takes the sequence of question embeddings $e = [e^{(1)}, \ldots, e^{(j)}]$ and produces the outputs $h = [h^{(1)}, \ldots, h^{(j)}]$ that are fed into the decoder's encoder-decoder attention layers. The decoder takes the sequence of response embeddings $l = [s, l^{(1)}, \ldots, l^{(j-1)}]$ and encoder's outputs h, producing the hidden vectors which are fed through the final linear layer to output the predicted dropout probabilities $\hat{y}_{DP} = [\hat{y}_{DP}^{(1)}, \ldots, \hat{y}_{DP}^{(j)}]$. Note that s is the starting token for the first position of the sequence. The overall process of DAS can be described as:

$$h = \text{Encoder}(e)$$
$$\hat{y}_{DP} = \sigma\big(\mathbf{W}_{\text{DP}}\text{Decoder}(s, l, h) + \mathbf{b}_{DP}\big)$$

where s is the start token embedding. The overall architecture of DAS is described in Figure 3.

We will now describe the components of each block in encoder and decoder. Each block mainly consists of a multi-head attention layer and a fully connected feed-forward layer. Multi-head attention network in each block takes queries, keys, and values of the sequence as inputs. Queries, keys, and values of $head_i$ are computed by multiplying weight matrices W_i^Q, W_i^K, W_i^V to the inputs as follows:

$$Q_i = e_Q W_i^Q = [Q_i^{(1)}, \ldots, Q_i^{(j)}]$$
$$K_i = e_K W_i^K = [K_i^{(1)}, \ldots, K_i^{(j)}]$$
$$V_i = e_V W_i^V = [V_i^{(1)}, \ldots, V_i^{(j)}]$$

Then, multi-head attention with h attention heads is computed as:

$$\text{Multihead}(e_Q, e_K, e_V) = \text{Concat}(head_1, \ldots, head_h)W^O$$
$$\text{where } head_i = \text{Softmax}(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i$$

d_k is the dimension of K_i, which is incorporated for scaling. W^O is the matrix to combine the outputs from multiple attention heads and to produce the final output of multi-head attention mechanism. Note that multi-head self attention uses same inputs to compute queries, keys and values while multi-head encoder-decoder attention uses outputs from the encoder as keys and values, which can be expressed as $Multihead(l, h, h)$. In order to prevent cheating from the future time steps, subsequent masks to the attention layers are incorporated. The fully connected feed-forward network applies linear transformation after adding non-linearity to the outputs of the multi-head attention layer as follows:

$$\text{FFN}(M) = \text{ReLU}(MW_1 + b_1)W_2 + b_2$$
$$\text{where } M = \text{Multihead}(e_Q, e_K, e_V)$$

Fine-Grained Versus Coarse-Grained Data for Estimating Time-on-Task in Learning Programming

Juho Leinonen
Aalto University
Espoo, Finland
juho.2.leinonen@aalto.fi

Francisco Enrique Vicente Castro
University of Massachusetts Amherst
Amherst, MA, USA
fcastro@cs.umass.edu

Arto Hellas
Aalto University
Espoo, Finland
arto.hellas@aalto.fi

ABSTRACT

The time that students spend on assignments, i.e. *time-on-task*, has been used frequently in prior research to understand student affect, study habits, and course performance, among others. The choice for how time-on-task is calculated, however, is typically based on available data. This data can be very coarse-grained, such as the timestamps from students' assignment submissions. Using coarse-grained data to calculate time-on-task has limitations, such as not being able to determine whether students take breaks when working on an assignment. In this work, we analyze the differences between two time-on-task metrics, one based on coarse-grained data—in this case, student submissions—and one based on fine-grained data—in this case, students' keystrokes during an assignment. We compare these two metrics and examine how well they correlate to find out whether time-on-task based on coarse-grained data can be an accurate metric for understanding the time spent by students on an assignment. Our results show that the correlation between the two metrics that are supposed to measure the same underlying phenomena—time-on-task—is only weak to moderate. This suggests that fine-grained data might be needed to accurately estimate time-on-task.

Keywords

time-on-task, fine-grained data, coarse-grained data, data granularity, keystroke data, programming process data, learning analytics, educational data mining

1. INTRODUCTION

Time-on-task—the amount of time that a student spends actively engaged in a task—is considered as one of the most important factors that contribute to learning and achievement [14, 30, 32]. Measuring time-on-task focuses on identifying *active time* that is spent on a task, instead of the overall time that includes breaks and time spent on unrelated activities. Time-on-task has been measured through various means: student self-reports [27], stopwatches [8], periodic observations [3], video recordings and eye movement data [4], and learning management system log data [17]. While all of these can be considered as proxies for time-on-task, accurately estimating time-on-task remains a challenging problem that deserves further attention [14, 17].

In this work, we study (a) to what extent two different types of log data—timestamped keystroke data and timestamped submission data from an introductory programming course—can be used to measure students' time-on-task and (b) to what extent time-on-task estimates produced with this data represent the same phenomenon. Our work is motivated by the need to distinguish between different types of data and the time-on-task estimates that can be produced with them. As numerous metrics have been used as proxies for time-on-task, if these metrics are not in line with each other, results from studies using them may not be comparable. That is, differences between observed results, or even contradictory results, could be explained to some extent by the difference in the chosen time-on-task metric.

Some studies similar to ours include work by Kovanović et al. [16] and Nguyen [23]. Kovanović et al. [16] built and compared a range of time-on-task metrics for evaluating students' performance, highlighting methodological issues. Nguyen [23], on the other hand, evaluated methods for identifying off-task behavior, also correlating the resulting estimates with academic performance. While these studies have used click-stream or event data from learning management systems such as Moodle, the data in our study comes from an introductory programming course where work on programming assignments is logged keystroke by keystroke.

This article is structured as follows. In Section 2, we discuss related time-on-task studies, starting with an overview of earlier studies on time-on-task and time-on-task estimates within learning programming, with a brief outline of studies that have analyzed different time-on-task estimates. We describe our context, data, research questions, and metrics in Section 3, and outline the analyses and results in Section 4. We discuss our findings and outline future work in Section 5.

2. RELATED WORK

2.1 Measuring Time-on-Task

Early work with time-on-task often involved on-site observations (e.g. in classrooms) where coders manually recorded

Juho Leinonen, Francisco Enrique Vicente Castro and Arto Hellas "Fine-Grained Versus Coarse-Grained Data for Estimating Time-on-Task in Learning Programming". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 648-653. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

and/or timed behaviors based on a coding rubric of on-task behaviors, also linking teacher's behavior with students' behavior (e.g. [2, 10, 15]). Over time, technology advancement led to the now-prevalent practice of mining *user interaction logs* from educational software used in classrooms or technology-augmented learning activities. This has made the analysis of user logs a vital component of more recent learning and behavior studies, such as in predicting help-seeking behavior [5] or assessing performance [9]. Time-on-task studies have likewise turned to this direction. Some examples (proxies for time-on-task in parentheses) include: analyzing the relationship of gamified elements to time-on-task (number of edits) [18] and comparing the impact of different course interventions to time-on-task (online interactions with peers and accessing course materials) [25]. Of note, both in earlier and more recent time-on-task studies, are the different measures or proxies used for time-on-task, and the methods used for identifying or approximating off-task activity and breaks. These are key factors that we explore in our comparison of coarse- and fine-grained time-on-task metrics.

In research focused on time-on-task in learning to program, a conventional approach has been to log user interactions within integrated development environments (see e.g. [13, 21, 22, 24, 29]). For example, Jadud [12] used the BlueJ IDE to capture code "snapshots" (copies of source code) whenever students compiled their programs, including compiler-reported errors and related metadata. Rodrigo et al. used BlueJ logs in combination with student surveys and observations to explore relationships between novice programmers' achievement, debugging, and syntax errors [26].

Submission data has been used to estimate students' *total elapsed time*, the total time between a student's first submission and last submission. Edwards et al. noted that the difference in total elapsed time between high- and low-scoring students is only small [6]. Similarly, the *time between compilation events* has been studied previously; Jadud observed that students are likely to recompile quickly after encountering a syntax error, but spend more time working on code after a successful compilation [11]. Definitions of *work sessions* also differ between studies. For example, Fenwick et al. [7] considered a "work session" terminated when no events were logged for 60 minutes. While the previous examples demonstrate the use of time from snapshots for estimating time-on-task, other studies in programming have explored using event counts (similar to other fields) for building predictive models of student achievement. For example, Ahadi et al. [1] used assignment-specific log data that included the number of "steps" that students took to solve each assignment for predicting course outcomes.

The time-on-task metrics in these (and other studies, e.g. [20, 28, 31]), however, suffer from similar problems of failing to capture the nuances around actual working time, as even the "work sessions" may fail to account for when and how students are working offline.

2.2 Analyzing Time-on-Task Estimates

Variations on time-on-task measures across studies and research instruments make it difficult to interpret and compare findings and bring into question whether or not the different metrics are indeed measuring or evaluating similar constructs. Some researchers have begun to explore this by looking at the different ways that researchers estimate time-on-task and analyzing how these estimation choices impact conclusions drawn from these measures.

Kovanović et al. [16], for example, looked at different time-on-task estimates from learning management system data and examined the impacts of these across courses from different subject domains. Their findings suggest that strategies for time-on-task estimation can have significant effects on learning analytics models of student performance. Using data collected from an introductory programming course, Leinonen et al. [19] examined a family of time-on-task-related metrics such as self-reported study time, log-based time spent on assignments, and event counts correlated with each other as well as course exam outcomes. They noted that while similar metrics such as edit counts and event counts tended to have higher correlations, exam scores were not strongly correlated with any of the metrics, except for the number of completed assignments.

While Leinonen et al. [19] did not analyze the impact of different break durations when estimating time spent on assignments, different break durations have been studied by both Kovanović et al. [16] and Nguyen [23]. Kovanović et al. and Nguyen both used time-on-task estimates based on timestamp differences between two subsequent events in learning management systems and highlight the importance of a good time-on-task estimation strategy.

Our work builds on this prior work by looking into data from an introductory programming course, where each keystroke associated with a course assignment was recorded and timestamped. Using this fine-grained log data, we study the impact of different thresholds for measuring off-task behavior, contrasting the keystroke data with submission-based data more commonly used in studies focusing on academic achievement in learning programming.

3. METHODOLOGY

3.1 Context and Data

The data for our study comes from a 7-week introductory programming course offered at a research first university in Europe. The workload of the course is 5 ECTS, which corresponds to roughly 100 to 125 study hours. In the course, students learn the basics of procedural and object-oriented programming in Java. The course uses a *many small assignments* approach, where many of the course assignments are small, but combine to form larger programs. After working on small assignments, students are given larger assignments as well, where they practice the content and constructs that they have learned earlier.

In total, the course had 147 programming assignments. The programming assignments are worked on in an integrated development environment (IDE), that logs keystroke data for plagiarism detection and research purposes. On each keystroke, the IDE collects the current timestamp and the modification to the source code of the assignment that the student is currently working on. Keystroke data is gathered only from course assignments. Additionally, information on when students submit their assignments is collected.

Students were informed about the data gathering on the course; our analyses included data from 137 students who consented to the use of their data for research purposes and who completed at least 10 assignments in the course.

3.2 Research Questions and Metrics

Our research questions are as follows:

RQ1. How do fine- and coarse-grained time-on-task metrics differ in terms of measuring time-on-task?

RQ2. Are there differences (a) between students and (b) between assignments on how well coarse-grained time-on-task correlates with fine-grained time-on-task?

In this study, we compare two different metrics for time-on-task that we call *coarse-grained time-on-task* and *fine-grained time-on-task*. The metrics are calculated for each student for each exercise they attempted and submitted.

Coarse-grained time-on-task is calculated as the difference between the timestamp of the first submission and the first keystroke event for that assignment. We used the first submission instead of the last submission since some students resubmitted assignments that they had previously completed "just in case" right before the deadline. However, the choice of first versus the last submission does not affect the results considerably: in 95% of the cases, students only had a single submission for each assignment.

Fine-grained time-on-task was calculated by computing the differences between keystroke timestamps in the data until the first submission of the assignment while ignoring any differences that were greater than a *break threshold* that is used to approximate off-task behavior or "outliers". Different values for the break threshold are explored and reported.

The key difference between the two metrics is that the fine-grained time-on-task takes into account the breaks that students take while working on assignments, whereas the coarse-grained time-on-task does not. If the break threshold is arbitrarily large, no breaks are removed when computing the fine-grained time-on-task, and the two metrics are identical.

4. ANALYSES AND RESULTS

4.1 Differences Between Time-on-Task Metrics

To answer *RQ1*, we first analyzed different break threshold values to examine how different thresholds affect the number of distinct study sessions in the data. We define a distinct study session as any sequence of snapshots for an assignment between breaks in the data, where what is considered as a break depends on the break threshold. We then examined how the choice of break threshold affects the correlation between the coarse- and fine-grained time-on-task metrics across the whole data set. The strength of the correlation between the metrics can signal whether the metrics are measuring the same phenomenon, i.e. time-on-task.

Figure 1, in Appendix, shows how having a different break threshold for the fine-grained time-on-task affects the number of distinct study sessions for thresholds between 30 seconds and 1200 seconds (i.e. 20 minutes). We see that having a very low threshold (e.g. anything under 100 seconds) results in a very high number of study sessions compared to having a higher break threshold (e.g. anything over 600 seconds, i.e. 10 minutes). The figure only shows the number of sessions up to a break threshold of 20 minutes since at that point, the decrease in the number of sessions is very small. What this essentially illustrates is that if a student takes a short break of under 200 seconds or so, they are quite likely to return to the task, but if the break is longer (e.g. over 10 minutes), they are not likely to return to the task soon. Based on this, in our data, a break threshold of around 600 seconds would seem reasonable as at that point, the rate of decrease plateaus.

Figure 2 (Appendix) shows the Pearson's correlation coefficient between coarse- and the fine-grained time-on-task metrics for different break thresholds between 30 seconds and 1200 seconds (20 mins.). We first note that for all the thresholds visualized in Figure 2, the correlation is weak since it varies between 0.33 and 0.37. The figure shows that the correlation increases slightly as the break threshold gets bigger, but similar to the number of study sessions, the rate of increase seems to plateau at around the 600 second (10 min.) mark. The correlation does continue increasing beyond what is visualized in the figure and eventually, at around 13 days, it reaches 1, where the fine- and coarse-grained time-on-task metrics are equal. This means that some students had a break of around 13 days within a single assignment.

4.2 Student and Assignment-Specific Correlations Between Time-on-Task Metrics

To answer *RQ2a*, we first calculated both time-on-task metrics for each student for each assignment they submitted. We then calculated the correlation between the metrics for each student separately, which leaves us with a single correlation per student. We examine the distribution of these correlations to understand if there are differences between students on how much the fine- and coarse-grained time-on-task metrics correlate. To answer *RQ2b*, we calculated the correlation between the coarse- and fine-grained metrics for each assignment separately, leaving us with a single correlation per assignment. Similar to *RQ2a*, we study the distribution of these correlations to see if there are assignment-specific differences in how well the two metrics correlate.

For analyzing student and assignment-specific differences in how well the coarse- and fine-grained time-on-task metrics correlate, we used a break threshold of 600 seconds (i.e. 10 minutes) for the fine-grained time-on-task metric. We chose 600 seconds as the results for *RQ1* showed that in our data, 600 seconds seems like a reasonable value to consider a student being on a break (Section 4.1).

Figure 3, in Appendix, shows the distribution of the correlations between the coarse- and fine-grained time-on-task metrics for individual students. The mean correlation is 0.47 with a standard deviation of 0.24 and the 95% confidence interval is 0.43 to 0.51. We notice from the figure that there are differences between students in how well the coarse- and fine-grained time-on-task metrics match each other. On average, the correlation seems moderate, with most students having a correlation between 0.2 and 0.6.

Figure 4, in Appendix, shows the distribution of the correlations between the coarse- and fine-grained time-on-task met-

rics for individual assignments. The mean correlation is 0.33 with a standard deviation of 0.21 and the 95% confidence interval is 0.30 to 0.36. We notice from the figure that similar to students, there are also differences between assignments. Compared to the between-students analysis (*RQ2a*), the assignment distribution is slightly more centered around the mean. Similar to the between-student analysis, the correlations for the assignments are also, on average, moderate.

5. DISCUSSION

5.1 Coarse- vs Fine-Grained Time-on-Task

We observed that our coarse-grained time-on-task metric poorly approximated our fine-grained time-on-task metric. The coarse-grained metric imitates metrics from earlier work where time-on-task has been calculated based on, for example, students' first and last submissions for an assignment [6], while the fine-grained time-on-task metric is somewhat similar to earlier works that utilized LMS trace data [16], although considerably more fine-grained.

We propose that the fine-grained metric explored in this work is a better metric for measuring time-on-task than a metric that relies on coarse-grained data, but removes outliers to keep time-on-task values meaningful. Prior work has suggested, for example, that large values are just ignored [16]. However, if we rely on removing outliers, we are bound to include data that is not accurate that was simply not caught by the outlier detection. For example, if two students both have a time-on-task estimate of two hours with a coarse-grained time-on-task metric, it is possible that one of them worked for ten minutes, while the other worked for a full 120 minutes. In this case, the actual time-on-task is drastically different, but the coarse-grained time-on-task estimate would be the same for both.

One downside of the fine-grained time-on-task metric is that it requires a break threshold to calculate time-on-task. Deciding on a good break threshold is not straightforward, and is most likely context-dependent. This work is not the first to note this issue: for example, both Nguyen [23] and Kovanović et al. [16] examined different cut-offs for outlier detection, which is similar to our work in examining different break thresholds.

5.2 Student- and Assignment-Specific Correlations

We identified student- and assignment-specific differences in how well coarse- and fine-grained time-on-task metrics correlate. This makes sense since the main difference between the metrics is that the fine-grained metric takes the breaks students take into account; thus, if a student does not take many breaks while working on assignments, the difference between the two time-on-task metrics will not be significant compared to a student who takes long breaks within single assignments. Here, factors such as possible previous programming experience and study fatigue may come into play and should be analyzed in future work.

Similarly, we found that there are differences between assignments in how much the two metrics correlate. Since the course has many small assignments, but also some bigger, more complex assignments, it makes sense that, for example, students might take more breaks during the bigger assignments compared to the smaller ones, which would have an effect on the correlation between the two metrics.

5.3 Conclusion and Future Work

In this work, we studied how two different time-on-task metrics built from programming log data correlate with each other. One of the metrics utilizes fine-grained keystroke data and takes the breaks students take during assignments into account by not including the breaks in its time-on-task estimate. The other time-on-task metric is more coarse-grained and includes any breaks students take during assignments in its time-on-task estimate.

Our results show that the correlation between the two metrics is at best moderate, which suggests that the choice of time-on-task metric can significantly impact the results of studies based on time-on-task analysis. This brings into question whether previous results that have used different metrics for measuring time-on-task are comparable with one another. Additionally, our results show that, at least in our context, there are also student- and assignment-specific differences in how much the two metrics correlate.

We acknowledge that we do not have a ground truth for time on task, i.e., both our metrics are only proxies. As part of our future work, we are looking into augmenting keystroke data from the programming environment with log data from other learning environments and self-reported time-on-task estimates. Similarly, in this work, we examined different break thresholds over all the data when identifying a break threshold; in future work, we will be looking at to what extent optimal break thresholds vary between students. We also acknowledge that we did not analyze how time-on-task relates to course outcomes, which has often been included in time-on-task studies (e.g. [16, 19, 23]). In the future, we will also be looking into how the studied metrics and different break thresholds relate to course performance.

6. REFERENCES

[1] A. Ahadi, R. Lister, H. Haapala, and A. Vihavainen. Exploring machine learning methods to automatically identify students in need of assistance. In *Proceedings of the eleventh annual international conference on international computing education research*, pages 121–130, 2015.

[2] L. W. Anderson and C. C. Scott. The Relationship Among Teaching Methods, Student Characteristics, and Student Involvement in Learning. *Journal of Teacher Education*, 29(3):52–57, May 1978. Publisher: SAGE Publications Inc.

[3] R. S. Baker, A. T. Corbett, K. R. Koedinger, and A. Z. Wagner. Off-task behavior in the cognitive tutor classroom: when students" game the system". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 383–390, 2004.

[4] C. Calderwood, P. L. Ackerman, and E. M. Conklin. What else do college students "do" while studying? an investigation of multitasking. *Computers & Education*, 75:19–29, 2014.

[5] F. E. V. Castro, S. Adjei, T. Colombo, and N. Heffernan. *Building Models to Predict*

Hint-or-Attempt Actions of Students. International Educational Data Mining Society, June 2015.

[6] S. H. Edwards, J. Snyder, M. A. Pérez-Quiñones, A. Allevato, D. Kim, and B. Tretola. Comparing effective and ineffective behaviors of student programmers. In *Proceedings of the fifth international workshop on Computing education research workshop*, pages 3–14, 2009.

[7] J. B. Fenwick Jr, C. Norris, F. E. Barry, J. Rountree, C. J. Spicer, and S. D. Cheek. Another look at the behaviors of novice programmers. *ACM SIGCSE Bulletin*, 41(1):296–300, 2009.

[8] S. Getting and K. Swainey. First graders with ipads?. *Learning & leading with technology*, 40(1):24–27, 2012.

[9] J. D. Gobert, M. S. Pedro, J. Raziuddin, and R. S. Baker. From Log Files to Assessment Metrics: Measuring Students' Science Inquiry Skills Using Educational Data Mining. *Journal of the Learning Sciences*, 22(4):521–563, Oct. 2013.

[10] T. L. Good and T. M. Beckerman. Time on Task: A Naturalistic Study in Sixth-Grade Classrooms. *The Elementary School Journal*, 78(3):193–201, Jan. 1978. Publisher: The University of Chicago Press.

[11] M. C. Jadud. A First Look at Novice Compilation Behaviour Using BlueJ. *Computer Science Education*, 15(1):25–40, Mar. 2005.

[12] M. C. Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the second international workshop on Computing education research*, pages 73–84, 2006.

[13] P. M. Johnson, H. Kou, J. Agustin, C. Chan, C. Moore, J. Miglani, S. Zhen, and W. E. Doane. Beyond the personal software process: Metrics collection and analysis for the differently disciplined. In *25th International Conf. on Software Engineering, 2003. Proceedings.*, pages 641–646. IEEE, 2003.

[14] N. Karweit. Time-on-task reconsidered: Synthesis of research on time and learning. *Educational leadership*, 41(8):32–35, 1984.

[15] J. S. Kounin and P. V. Gump. Signal systems of lesson settings and the task-related behavior of preschool children. *Journal of Educational Psychology*, 66(4):554–562, 1974. Place: US Publisher: American Psychological Association.

[16] V. Kovanović, D. Gašević, S. Dawson, S. Joksimović, R. S. Baker, and M. Hatala. Does time-on-task estimation matter? implications for the validity of learning analytics findings. *Journal of Learning Analytics*, 2(3):81–116, 2015.

[17] V. Kovanović, D. Gašević, S. Dawson, S. Joksimović, R. S. Baker, and M. Hatala. Penetrating the black box of time-on-task estimation. In *Proceedings of the fifth international conference on learning analytics and knowledge*, pages 184–193, 2015.

[18] R. N. Landers and A. K. Landers. An Empirical Test of the Theory of Gamified Learning: The Effect of Leaderboards on Time-on-Task and Academic Performance. *Simulation & Gaming*, 45(6):769–785, Dec. 2014. Publisher: SAGE Publications Inc.

[19] J. Leinonen, L. Leppänen, P. Ihantola, and A. Hellas. Comparison of time metrics in programming. In *Proceedings of the 2017 ACM conf. on International Computing Education Research*, pages 200–208, 2017.

[20] L. Leppänen, J. Leinonen, and A. Hellas. Pauses and spacing in learning to program. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, pages 41–50, 2016.

[21] J. McKeogh and C. Exton. Eclipse plug-in to monitor the programmer behaviour. In *Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange*, pages 93–97, 2004.

[22] C. Murphy, G. Kaiser, K. Loveland, and S. Hasan. Retina: helping students and instructors based on observed programming activities. In *Proceedings of the 40th ACM technical symposium on Computer Science Education*, pages 178–182, 2009.

[23] Q. Nguyen. Rethinking time-on-task estimation with outlier detection accounting for individual, time, and task differences. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 376–381, 2020.

[24] C. Norris, F. Barry, J. B. Fenwick Jr, K. Reid, and J. Rountree. Clockit: collecting quantitative data on how beginning software developers really work. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 37–41, 2008.

[25] S. Park. Analysis of Time-on-Task, Behavior Experiences, and Performance in Two Online Courses with Different Authentic Learning Tasks. *International Review of Research in Open and Distributed Learning*, 18(2):213–233, 2017. Publisher: Athabasca University Press (AU Press).

[26] M. M. T. Rodrigo, T. C. S. Andallaza, F. E. V. G. Castro, M. L. V. Armenta, T. T. Dy, and M. C. Jadud. An Analysis of Java Programming Behaviors, Affect, Perceptions, and Syntax Errors among Low-Achieving, Average, and High-Achieving Novice Programmers. *Journal of Educational Computing Research*, 49(3):293–325, Oct. 2013.

[27] M. Romero and E. Barbera. Quality of learners' time and learning performance beyond quantitative time-on-task. *International Review of Research in Open and Distributed Learning*, 12(5):125–137, 2011.

[28] J. Spacco, P. Denny, B. Richards, D. Babcock, D. Hovemeyer, J. Moscola, and R. Duvall. Analyzing student work patterns using programming exercise data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 18–23, 2015.

[29] J. Spacco, D. Hovemeyer, W. Pugh, F. Emad, J. K. Hollingsworth, and N. Padua-Perez. Experiences with marmoset: designing and using an advanced submission and testing system for programming courses. *ACM Sigcse Bulletin*, 38(3):13–17, 2006.

[30] J. Stallings. Allocated academic learning time revisited, or beyond time on task. *Educational researcher*, 9(11):11–16, 1980.

[31] E. S. Tabanao, M. M. T. Rodrigo, and M. C. Jadud. Identifying at-risk novice java programmers through the analysis of online protocols. In *Philippine Computing Science Congress*, pages 1–8, 2008.

[32] H. J. Walberg. Synthesis of research on time and learning. *Educational leadership*, 45(6):76–85, 1988.

APPENDIX
A. FIGURES

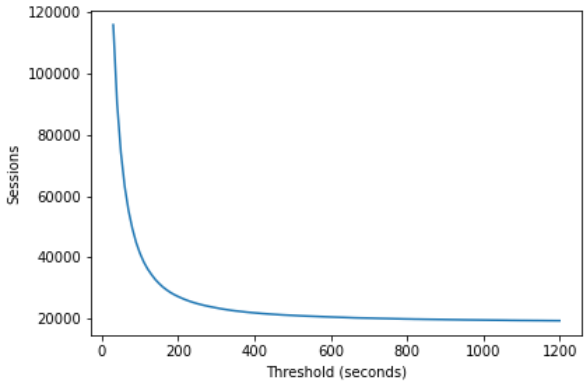


Figure 1: Number of distinct study sessions with different thresholds for breaks for the fine-grained time-on-task metric. The x-axis is the threshold for considering the student to be on a break in seconds. The y-axis is the number of study sessions in the data. Data is shown for thresholds between 30 seconds and 1200 seconds (20 minutes).

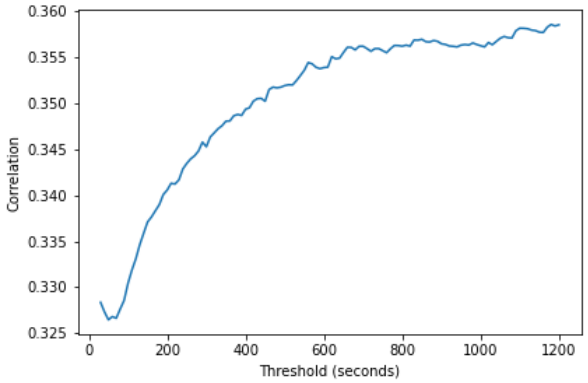


Figure 2: The correlation between the coarse and the fine-grained time-on-task metric with different thresholds for breaks for the fine-grained time-on-task metric. The x-axis is the threshold for considering the student to be on a break in seconds. The y-axis is the Pearson correlation coefficient between the coarse and the fine-grained time-on-task metric. Data is shown for thresholds between 30 seconds and 1200 seconds (20 minutes).

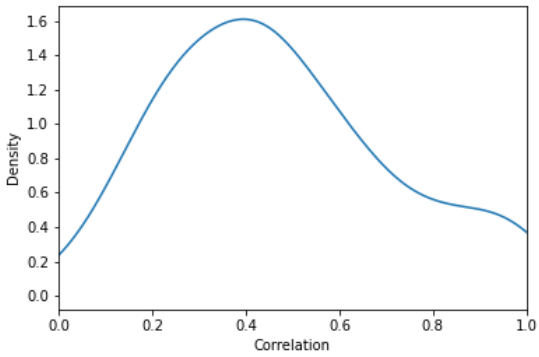


Figure 3: The distribution of student-specific correlations between the coarse- and fine-grained time-on-task metrics, where fine-grained time-on-task was calculated with a 600 second break threshold. The x-axis is the Pearson correlation coefficient and the y-axis is the density.

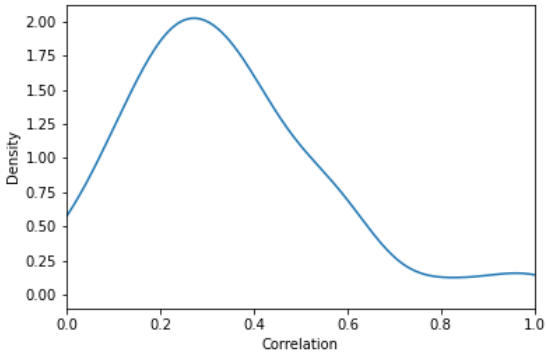


Figure 4: The distribution of assignment-specific correlations between the coarse and the fine-grained time-on-task metrics, where fine-grained time-on-task was calculated with a 600 second break threshold. The x-axis is the Pearson correlation coefficient and the y-axis is the density.

The Impact of Learning Analytics on Student Performance and Satisfaction in a Higher Education Course

Dimitrios Tzimas, Stavros Demetriadis

Aristotle University of Thessaloniki
{detzimas, sdemetri}@csd.auth.gr

ABSTRACT

Learning analytics (LA) is collecting, processing, and visualization of big data to optimize learning. This article aims to interpret the impact of analyzing learning data for tertiary education. The article describes a semester-long mixed methods study for 63 students enrolled in a Greek technical university laboratory, retrieving data from the learning management system (LMS). We applied minimal LA guidance in the experimental group and no LA guidance in the control group. The research questions are as follows: Can a student-facing learning analytics approach at minimal level guidance improve students' LMS access and learning performance levels? Are the students' LMS access, discussion forums, and submitted assignments, critical predictors for students' course grades? What are students' opinions about learning analytics as a tool for data-driven decision-making strategy? The study followed the do-analyze-change-reflect LA model. The data collected included students' time spent on LMS, exercises, and discussion posts, while the dependent variable was the course grade. Results indicate that it increased the students' LMS access and satisfaction when we applied LA but not their final grade. Future research could apply higher effort interventions and stronger teacher guidance to provide insights into student performance, engagement, and satisfaction.

Keywords

Student-facing learning analytics, Performance, Satisfaction, Teacher guidance, Post-secondary education.

1. INTRODUCTION

Learning analytics is a multidisciplinary field between computer science and education that fosters the learning process based on big data monitoring [10]. In [29], the authors defined LA as the measurement, analysis and reporting of data about learners and their contexts, for purposes of optimizing learning and the environments in which it occurs. Furthermore, the LA tasks are a set of handy tools to collect and analyze the data accumulated in a smart classroom for data-based decision-making [1]. Consequently, without analytics, instructors cannot provide guidance at appropriate times when students encounter difficulties [11]. In parallel, institutions have embedded LA techniques to enhance retention rates, use resources effectively, and increase students' engagement, satisfaction, and motivation [26].

The authors conducted this mixed-methods study with the research objective of mapping student-facing learning analytics (LA) in real tertiary educational settings. The article is organized as follows: (1) we conduct a short literature review, (2) we explain how the research questions were formulated, (3) we illustrate the design and results of the experiment, and (4) we present the discussion and conclusions reached.

2. RELATED WORK

Student-facing LA is a subfield of LA and focuses on the reporting phase, such as LA dashboards, educational recommender and feedback systems [5, 6]. It is challenging to show students a dashboard or automated emailing systems and conduct surveys to extract usage insights [20]. According to [20], a well-established student-facing LA system consists of four learning design phases (do-analyze-change-reflect). To provide a theoretical framework and extract the research questions, we conducted a short literature review about student-facing LA. The studies can be classified in terms of (1) improvement of performance, (2) prediction of student course grade, (3) improvement of LMS access, and (4) student opinions and satisfaction of LA.

A series of studies [23, 30] have explored the idea of student-facing LA improving levels of performance. Students' final marks could determine the assessment of their academic achievement [19, 33]. In contrast, academic performance and attainment are not related to student access behavior perforce [17]. Nevertheless, we argue that only a few studies examine under LA interventions the correlations between LMS use, the number of submitted assignments, and forum posts as metrics for performance. Furthermore, we need more research to examine if the low effort LA interventions could positively affect students' performance. After all, explaining the students' learning performance is a continual research question.

LA predictive modeling is a core practice of scholars focusing on student success [22]. In [18], a data mining process constructs variables that reflect the theoretical evidence and measure a prediction model's accuracy. In addition, [31] presented a prediction model for failure-prone students using neural networks techniques. These studies emphasize that student-performance prediction is a dominant research domain. Despite the above studies, we argue that building a predictive model for students' performance based on critical predictors such as LMS participation and submitted assignments is an interesting research question.

Engagement can substantially impact students' performance [4, 14]. In [6], the authors have explored the idea of student-facing LA, improving levels of engagement. They have indicated that academic engagement is a multi-dimensional construct and refers to students' level of involvement [8, 15]. However, we argue that not many studies examine the effect of student-facing LA interventions on students' level of engagement.

Dimitrios Tzimas and Stavros Demetriadis "The Impact of Learning Analytics on Student Performance and Satisfaction in a Higher Education Course". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 654-660. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Targeted studies exist on students' opinions of LA [20, 28]. The [32] study empirically explored the effects of a mobile LA tool in student satisfaction. Nevertheless, we consider that students' opinions before and after LA interventions need further research and could extract valuable insights concerning students' satisfaction and expectations. The surveys' results will confirm or not the existing ones.

Drawing upon the findings of the above studies, our purpose is to investigate the open issues. It would be meaningful to know whether subjects in the feedback conditions gain learning benefits such as performance and satisfaction. In parallel, integrating the LA concepts into tertiary classroom practice has been slow [12]. This article replicates similar research and aims to interpret analyzing learning data for higher education institutions (HEI).

2.1 Research Questions

Within this context, the current experimentation study poses the following research questions:

1. Can a student-facing learning analytics approach at minimal level guidance improve students' LMS access and learning performance levels?

2. Are the students' LMS access, discussion forums, and submitted assignments, critical predictors for students' course grades?

3. What are students' opinions about learning analytics as a tool for data-driven decision-making strategy?

3. METHOD

3.1 Participants and Context

This study took place in the authentic context of a sixth-semester 13-week undergraduate department laboratory course, "digital signal processing" (DSP), at a Greek HEI computer science department between February 2018 and June 2018. The reason for selecting this particular blended course was the high dropout and failure rate in the past exams. This study focused on 31 students as an experimental group receiving the LA intervention ("treatment") with minimal teacher guidance tested for comparison purposes. Participants were 26% female. The control group had 32 students who received no particular LA intervention. The instructor had two-hour lectures and face-to-face meetings/office hours on Mondays every week with the students.

An overview of the LA tool that students used follows: The Open eClass platform is an open-source LMS and is developed by the non-profit civil company called "Greek Universities Network" (GUNET) (https://www.gunet.gr/en/). The platform's main features follow: Management of electronic courses and educational content; Student management; Information, communication, collaboration, evaluation and feedback tools. The structure of the course was as follows:

Week 1. Module 1: The nature of DSP was explored. To ensure transparency and institution-wide adoption [34], we informed the department principal in detail about the experiment, after which she enthusiastically gave her consent. It was then defined what types of data should be tracked and that the feedback (dashboard and messages) would be intended for students.

Week 2. Module 2: Fundamental signals. The first coding exercise was performed in addition to weekly discussion threads and office hours. We gave a detailed description of which student-facing LA will be used and how students will utilize them.

Week 3. Module 3: Digital signal sampling. For usability testing, the students described their initial experience of using LA. The students were surprised, as many claimed that it was impossible to support concepts such as monitoring, analyzing, and feedback.

Week 4. The first quiz assignment and second coding exercise took place. The instructor contributed to the discussion forum to give a sense of learning community. We provided verbal encouragement for students to access their statistics and figures via the LA tool to reflect and meditate.

Week 5. The second quiz assignment and third coding exercise took place—module 4: Fourier transformation principles. We discussed the self-reflection and meditation process.

Week 6. Active intervention and feedback with personalized messages containing the grades of the students' assignments, recommendations, and comparisons of their performance with aggregated data (e.g., participation in discussions and submission of assignments). The encouraging wording of the messages was designed to benefit pedagogically and not harm the student. For instance, "do you need some support?" or "you could participate more in the discussion forum." We provided personalized feedback with visualizations for tracking students' learning progress.

Week 8 and 9. Module 5: Digital filters. Provide in-class feedback (figure 1), recommendations, and scheduling for personalized scaffolding. Verbal suggestions informed students about what to do based on analytics.

Week 10 and 11. The third quiz and an exercise took place. We provided in-class information about absences, participation, and homework. Students received personalized messages with visualizations of their learning progress for mirroring, self-reflection, and motivation.

Week 12 and 13. A revision session and a collaborative quiz were conducted in addition to weekly monitoring and analysis. We used a think-aloud protocol to understand how students reclaimed feedback. A final questionnaire took place—Week 14. The final examination was conducted.

Figure 1. Personalized feedback with visualizations for mirroring.

3.2 Research Design

A mixed quantitative and qualitative method case study was utilized to provide an instantiation of the LA framework with a description of the methodology for others to use a similar process. To answer the first and second research questions, we conducted a causal research design.

3.3 Measures and Instruments

The target of our intervention is students' LMS access, performance, and learning satisfaction. The analysis object is discussion forum use, the number of exercises submissions, and LMS access, while all variables are numeric.

Performance: The performance is measured by a simple dependent variable, the final course grade, that has convenient properties for causal and statistical analysis. The grading system of the final exam is as follows: Scale: 0.00 – 10.00 / Pass: 5:00 (excellent: 8.50 – 10.00 / very good: 6.5 – 8.49 / good: 5.00 – 6.49). The independent scale variables and their definitions that we have considered were: "discussions" that counts the number of posts per student, thus the LMS discussion forum's involvement; "exercises" that counts the submitted assignments accumulated. Each weekly assignment asked true/false, multiple-choice, open-ended questions, and coding exercises; "hoursonlms" that counts, in hours, students' LMS access.

LMS access: Student engagement is a complicated measured construct but vital for students' success that encloses more than participation, motivation, and self-regulation [21]. Therefore, student LMS access in time is an indicator of student engagement.

Satisfaction: The instruments that we used to collect student opinion data are two student opinion questionnaires. An individual questionnaire was administered at the beginning of the course and another at the end. The questionnaire data incorporated participants' reflections on the activity and helped us to collect qualitative data about their opinions as an evaluation.

3.4 Data Collection and Analysis

The level of significance was set at p = 0.05. Graphically, we examined the same assumptions and checked for no outliers. Some visualizations (i.e., dot plot, histogram, a boxplot for density, skewness, and variability) were produced. Finally, for data processing and analysis, the SPSS 25.0 statistical application processed the data.

4. RESULTS

We first applied normality (Shapiro-Wilks) and variance (Levene) controls on available data. The results ($p > 0.05$) indicated statistical non-significance suggesting that sample data come from normal distributions and populations with the same variance, therefore appropriate for parametric test analysis.

4.1 Research Question 1

Hypothesis 1: The performance, as measured by the course grade of the experimental group, is not statistically significantly different from that of the control group.

The mean score of the experimental group (M = 6.08, SD = 2.62) was slightly higher than that of the control group (M = 5.49, SD = 1.60). However, the independent samples t-test comparing course grades between the groups revealed no statistically significant differences ($t = 1.077$, $p = 0.287$) (Table 2). Overall, the null hypothesis failed to be rejected.

Table 2. The t-test results of the experimental and control groups for performance

Group	N	Mean	SD	t	p
Experimental	31	6.08	2.62	1.077	0.287
Control	32	5.49	1.60		

Hypothesis 2: The experimental group's LMS access (in hours) is not statistically significantly different from that of the control group.

Table 3 shows that the mean of the overall LMS access for the experimental group was higher than that of the control group. The independent samples t-test comparing LMS use between the control and experimental groups revealed statistically significant differences ($t = 4.610$, $p = 0.000$). Overall, the null hypothesis is rejected.

Table 3: The t-test results for LMS access

Group	N	Mean	SD	t	p
Experimental	31	10.03	7.79	4.610	0.000
Control	32	3.41	1.84		

4.2 Research Question 2

Focusing on the experimental group, we examined the Pearson correlations (Table 4), extracting that the submitted exercises ("exercises") are highly positively correlated with the final course grade ("finalgrade"). Also, time spent on LMS ("hoursonlms") is weakly positively correlated with the final course grade. However, there is a tendency but no statistically significant correlation between forum posts ("discussions") and the final course grade.

Afterward, a simple regression analysis was conducted for "exercises" to estimate the final grade. The check (ANOVA) of the hypothesis that no regression showed that this hypothesis is rejected (F = 18.156, p = 0.000). To evaluate this regression model, the Pearson correlation coefficient (Table 4) (R = 0.620, $p = 0.000$) reflects the predictor importance; thus, we extracted a good predictor. Then, the model accuracy (quality) is 61.1% and the determination factor (R-squared = 0.385 < 0.5) is considered a low effect size. Finally, the model's equation is y=1.002*x+3.954 (y: final grade, x: exercises).

Then, a simple regression analysis was conducted for "hoursonlms" to estimate the final grade. The Pearson correlation coefficient (Table 4) (R = 0.392, $p = 0.015$) reflects the predictor importance. Thus, we extracted a weak predictor with a determination factor (R-squared = 0.154 < 0.3) to be considered a weak effect size. Furthermore, we observe a high correlation (R = 0.749, $p = 0.000$) between the above two predictors. As a result, we decided that there is no need to conduct a multiple regression analysis.

Table 4. Pearson correlations, in parentheses Sig. (two-tailed)*

	finalgrade	hoursonlms	exercises	discussions
finalgrade	1.000	0.392 (0.015)	0.620 (0.000)	0.284 (0.061)
hoursonlms		1.000	0.749 (0.000)	0.525 (0.001)
exercises			1.000	0.314 (0.043)

*Significant difference at the 0.05 level

4.3 Research Question 3

We applied two questionnaires to address the third research question (What are students' opinions about learning analytics as a tool for data-driven decision-making strategy?). Twenty-three students submitted the first questionnaire (appendix), and the purpose was to determine as a baseline their prior knowledge of the LA field. Fifteen students submitted the final questionnaire. The purpose was to determine their thoughts about the LA experience and their overall satisfaction and acceptance. Students gave responses in the free comments field. Based on the mining of students' opinions and perceptions, the LA experience increased students' learning satisfaction. To summarize, students argue that: LA feedback is helpful for their learning progress; they expect that LA is applied in most courses; student-facing LA tools via smartphone would have an added-value impact; peer-comparison progress dashboards increase their engagement.

5. DISCUSSION AND CONCLUSIONS

5.1 Research Question 1

Table 3 shows that LMS access was significantly higher for the treatment group who used LA. This result is consistent with that mentioned by [3] and [25]. Students were triggered using LA to submit more quizzes and exercises (cognitive activities). In addition, students increased their sense of belonging to an online community. However, the findings suggest that high LMS access does not necessarily affect performance [36].

The experimental group had slightly higher scores than the control group (Table 2). This result is consistent with the one mentioned by [27], who stated that "students valued the information, but, despite high engagement with the information, students' study behavior and learning outcome remained rather unaffected." In contrast, many studies [3, 13, 16, 24, 25] have stated that students tend to perform better when the students accept LA interventions. An explanation is that our LA approach resulted in delayed and low effort interventions, which affected the students' overall performance. The standard deviation (SD) values in Table 2 show high diaspora, especially in the experimental group, so we argue that the LA impact affected the students in an outspread way. We conclude that there is no performance improvement without the instructor's strong guidance and targeted interventions.

5.2 Research Question 2

Some of our findings are consistent with the results of other related studies. Based on Table 4, we observe moderate statistical correlations between time spent on LMS and the final grade and between the number of assignments' submissions and the final course grade. This result is aligned with that mentioned by [2] and [15]. Our prediction model for academic performance confirms the results of related studies; thus, we need models with higher accuracy and effect size [7, 9].

5.3 Research Question 3

We conclude that the students' satisfaction was high, in agreement with findings in [25]. Students' positive response to the usefulness of student-facing LA is in agreement with the literature [6, 30]. In addition, the students' responses in the reflection phase confirm the discussion of [20] that students should analyze their behavior using their self-regulated methods. In accordance with [35], the above findings strengthen understanding students' opinions of LA qualitatively rather than as technical methods. Furthermore, interpreting students' comments, we argue that students liked this new learning approach following personalized reports. Students would like LA personalized interventions with a smartphone application and comparisons of their learning progress with their classmates. In conclusion, students' sample quotes extract emerging themes: awareness of others in the class, motivation, increased satisfaction and self-regulation, and technical proposals.

5.4 Limitations

We acknowledge that there are certain limitations to this small-scale study that prevent its findings from being generalized. First, the small sample size and the context of the dataset limit the findings. The data covers one semester on a very domain-specific course at one Greek university, and institutional factors influence the results. Furthermore, the LMS captures a subset of all the events in a learning experience, while other student characteristics may influence student outcomes. It would be useful to search for other factors or latent variables that might differ between the two groups in order to improve the results. Second, engagement was measured in terms of quantity rather than quality. More factors that influence student engagement quality should be studied, such as teacher participation and student effort. Third, the questionnaire's answers indicate that students in the experimental group are satisfied with the LA tool; however, we do not know how LA impacts students' decision-making strategy.

5.5 Future work

It is our intention to replicate the study with another treatment group applying a strong (high effort) teacher guidance to see the impact in relation to the minimal (low effort) group. We will evaluate the impact of three levels of LA interventions: mirroring, metacognitive activities, and explicit guidance. Furthermore, we intend to focus on replicating the experiment in other course settings with larger populations, different profiles, and the use of a mobile-based user-centered LA application. It would be constructive to build and test a predictive model with higher accuracy and stronger effect size applying sophisticated machine learning or deep learning algorithms.

6. REFERENCES

[1] Aguilar, J., Sánchez, M., Cordero, J., Valdiviezo-Díaz, P., Barba-Guamán, L., & Chamba-Eras, L. (2017). Learning analytics tasks as services in smart classrooms. *Universal Access in the Information Society*, 1–17. https://doi.org/10.1007/s10209-017-0525-0.

[2] Akhtar, S., Warburton, S., & Xu, W. (2017). The use of an online learning and teaching system for monitoring computer aided design student participation and predicting student success. *International Journal of Technology and Design Education*, 27(2), 251–270. https://doi.org/10.1007/s10798-015-9346-8.

[3] Atherton, M., Shah, M., Vazquez, J., Griffiths, Z., Jackson, B., & Burgess, C. (2017). Using learning analytics to assess student engagement and academic outcomes in open access enabling programmes. *Open Learning: The Journal of Open, Distance and E-Learning*, 32(2), 119–136. https://doi.org/10.1080/02680513.2017.1309646.

[4] Blasco-Arcas, L., Buil, I., Hernández-Ortega, B., & Sese, F. J. (2013). Using clickers in class. The role of interactivity, active collaborative learning and engagement in learning performance. *Computers and Education*, 62, 102–110. https://doi.org/10.1016/j.compedu.2012.10.019.

[5] Bodily, R., Ikahihifo, T. K., Mackley, B., & Graham, C. R. (2018). The design, development, and implementation of student-facing learning analytics dashboards. *Journal of Computing in Higher Education*, 30(3), 572–598. https://doi.org/10.1007/s12528-018-9186-0.

[6] Bodily, R., & Verbert, K. (2017). Trends and issues in student-facing learning analytics reporting systems research. *ACM International Conference Proceeding Series*, 309–318. https://doi.org/10.1145/3027385.3027403.

[7] Casey, K., & Azcona, D. (2017). Utilizing student activity patterns to predict performance. *International Journal of Educational Technology in Higher Education*, 14(1). https://doi.org/10.1186/s41239-017-0044-3.

[8] Chen, P. S. D., Lambert, A. D., & Guidry, K. R. (2010). Engaging online learners: The impact of Web-based learning technology on college student engagement. *Computers and Education*, 54(4), 1222–1232. https://doi.org/10.1016/j.compedu.2009.11.008.

[9] Cohen, A. (2017). Analysis of student activity in web-supported courses as a tool for predicting dropout. *Educational Technology Research and Development*, 65(5), 1285–1304. https://doi.org/10.1007/s11423-017-9524-3.

[10] Crespo García, R.M., Pardo, A., Delgado Kloos, C., Niemann, K., Scheffel, M., & Wolpers, M. (2012). Peeking into the black box: visualizing learning activities. International Journal of Technology Enhanced Learning, Vol. 4, Nos. 1/2, pp.99–120. http://dx.doi.org/10.1504/IJTEL.2012.048313.

[11] Er, E., Gómez-Sánchez, E., Dimitriadis, Y., Bote-Lorenzo, M. L., Asensio-Pérez, J. I., & Álvarez-Álvarez, S. (2019). Aligning learning design and learning analytics through instructor involvement: a MOOC case study. *Interactive Learning Environments*, 27(5–6), 685–698. https://doi.org/10.1080/10494820.2019.1610455.

[12] Ferguson, R., Brasher, A., Clow, D., Cooper, A., Hillaire, G., Mittelmeier, J., ... Vuorikari, R. (2016). Research evidence on the use of learning analytics: Implications for education policy (Joint Research CentreScience for Policy Report, EUR 28294 EN). Luxembourg City: Publications Office of the European Union. https://doi.org/10.2791/955210.

[13] Firat, M. (2016). Determining the effects of LMS learning behaviors on academic achievement in a learning analytic perspective. *Journal of Information Technology Education: Research*, 15(15), 75–87. https://doi.org/10.28945/3405.

[14] Henrie, C. R., Halverson, L. R., & Graham, C. R. (2015). Measuring student engagement in technology-mediated learning: A review. *Computers and Education*, 90, 36–53. https://doi.org/10.1016/j.compedu.2015.09.005.

[15] Joksimović, S., Gašević, D., Loughin, T. M., Kovanović, V., & Hatala, M. (2015). Learning at distance: Effects of interaction traces on academic achievement. *Computers and Education*, 87, 204–217. https://doi.org/10.1016/j.compedu.2015.07.002.

[16] Khalil, M., & Ebner, M. (2017). Clustering patterns of engagement in Massive Open Online Courses (MOOCs): the use of learning analytics to reveal student categories. *Journal of Computing in Higher Education*, 29(1), 114–132. https://doi.org/10.1007/s12528-016-9126-9.

[17] Khan, T. M., Clear, F., & Sajadi, S. S. (2012). The relationship between educational performance and online access routines: Analysis of students' access to an online discussion forum. *ACM International Conference Proceeding Series*, May, 226–229. https://doi.org/10.1145/2330601.2330655.

[18] Kim, D., Park, Y., Yoon, M., & Jo, I. H. (2016). Toward evidence-based learning analytics: Using proxy variables to improve asynchronous online discussion environments. *Internet and Higher Education*, 30, 30–43. https://doi.org/10.1016/j.iheduc.2016.03.002.

[19] Kim, J., Jo, I. H., & Park, Y. (2016). Effects of learning analytics dashboard: analyzing the relations among dashboard utilization, satisfaction, and learning achievement. *Asia Pacific Education Review*, 17(1), 13–24. https://doi.org/10.1007/s12564-015-9403-8.

[20] Kitto, K., Lupton, M., Davis, K., & Waters, Z. (2017). Designing for student-facing learning analytics. *Australasian Journal of Educational Technology*, 33(5), 152–168. https://doi.org/10.14742/ajet.3607.

[21] Lacave, C., Velázquez-Iturbide, J. Á., Paredes-Velasco, M., & Molina, A. I. (2020). Analyzing the influence of a visualization system on students' emotions: An empirical case study. *Computers and Education*, 149(May 2019). https://doi.org/10.1016/j.compedu.2020.103817.

[22] Lang, C., Siemens, G., Wise, A., & Gasevic, D. (2017). Handbook of Learning Analytics. https://doi.org/10.18608/hla17.

[23] Li, Q., & Baker, R. (2018). The different relationships between engagement and outcomes across participant subgroups in Massive Open Online Courses. *Computers and Education*, 127(April 2017), 41–65. https://doi.org/10.1016/j.compedu.2018.08.005.

[24] Lu, O. H. T., Huang, J. C. H., Huang, A. Y. Q., & Yang, S. J. H. (2017). Applying learning analytics for improving students engagement and learning outcomes in a MOOCs enabled collaborative programming course. *Interactive Learning Environments*, 25(2), 220–234. https://doi.org/10.1080/10494820.2016.1278391.

[25] Nguyen, Q., Rienties, B., Toetenel, L., Ferguson, R., & Whitelock, D. (2017). Examining the designs of computer-based assessment and its impact on student engagement, satisfaction, and pass rates. *Computers in Human Behavior*, 76, 703–714. https://doi.org/10.1016/j.chb.2017.03.028.

[26] Olmos, M., & Corrin, L. (2012). Learning analytics: A case study of the process of design of visualizations. *Journal of Asynchronous Learning Network*, 16(3), 39–49. https://doi.org/10.24059/olj.v16i3.273.

[27] Ott, C., Robins, A., Haden, P., & Shephard, K. (2015). Illustrating performance indicators and course characteristics to support students' self-regulated learning in CS1. *Computer Science Education*, 25(2), 174–198. https://doi.org/10.1080/08993408.2015.1033129.

[28] Ruipérez-Valiente, J. A., Muñoz-Merino, P. J., Leony, D., & Delgado Kloos, C. (2015). ALAS-KA: A learning analytics extension for better understanding the learning process in the Khan Academy platform. *Computers in Human Behavior*, 47, 139–148. https://doi.org/10.1016/j.chb.2014.07.002.

[29] Siemens, G., & Long, P. (2011). Penetrating the fog: Analytics in learning and education. *EDUCAUSE Review*, 46(5), 30–40. Retrieved from https://er.educause.edu/articles/2011/9/penetrating-the-fog-analytics-in -learning-and-education.

[30] Smith, P. (2019). Engaging online students through peer-comparison progress dashboards. *Journal of Applied Research in Higher Education*, 12(1), 38–56. https://doi.org/10.1108/JARHE-11-2018-0249.

[31] Sukhbaatar, O., Usagawa, T., & Choimaa, L. (2019). An artificial neural network based early prediction of failure-prone students in blended learning course. *International Journal of Emerging Technologies in Learning*, 14(19), 77–92. https://doi.org/10.3991/ijet.v14i19.10366.

[32] Tabuenca, B., Kalz, M., Drachsler, H., & Specht, M. (2015). Time will tell: The role of mobile learning analytics in self-regulated learning. *Computers and Education*, 89, 53–74. https://doi.org/10.1016/j.compedu.2015.08.004.

[33] Timmers, C. F., Walraven, A., & Veldkamp, B. P. (2015). The effect of regulation feedback in a computer-based formative assessment on information problem solving. *Computers and Education*, 87, 1–9. https://doi.org/10.1016/j.compedu.2015.03.012.

[34] Tzimas, D., Demetriadis, S. (2021). Ethical issues in learning analytics: a review of the field. *Education Technology Research and Development*. https://doi.org/10.1007/s11423-021-09977-4.

[35] Viberg, O., Hatakka, M., Bälter, O., & Mavroudi, A. (2018). The current landscape of learning analytics in higher education. *Computers in Human Behavior*, 89(August), 98–110. https://doi.org/10.1016/j.chb.2018.07.027.

[36] Zingaro, D. (2014). Peer instruction contributes to self-efficacy in CS1. *In Proceedings of the 45th ACM technical symposium on computer science education* – SIGCSE'14 (pp. 373–378). New York, NY: ACM.

Appendix

First questionnaire

Question	Answer		
	Yes	**No**	**No answer**
Do you know what analytics is?	10	13	-
Do you know what learning analytics is?	6	17	-
Do you believe that the collection and processing of your learning data and behavior will be helpful to your learning experience?	22	1	-
Would you be interested in being informed about your learning progress concerning your classmates?	13	6	4
Would it be helpful to have feedback (e.g., personalized monitoring and individualized learning material) on your learning progress?	22	1	-

Final questionnaire

Question	Answer		
	Yes	**No**	**No answer**
Were the personalized notifications about your engagement, absences, and performance useful for the course?	14	1	-
Would you prefer more detailed information?	4	8	3
Would you prefer to receive notifications and messages through a smartphone?	10	2	3
Is the comparison of your learning progress with that of your classmates useful?	11	2	2
Please provide free comments: "It is the first time for me that a teacher has sent personalized messages to all students about their learning progress. I have nothing more to suggest. It would be great to convince the other teachers to do the same". "The whole procedure with the exercises, the open discussions, and generally the lecturers' teaching methods helped me very much to self-regulate. I enjoyed both class time and homework". "It was the first time that we had received such refined, analytical, and informed monitoring about our progress and performance on a course." "I would like access to an LA android-based application." "I liked the quizzes the most, and I would prefer to be informed via a smartphone app." "There was sufficient and motivational guidance from the instructor about online exercises." "Instructor's comments about the exercises on LMS were constructive, analytical, and motivational." "I would like more teacher guidance about the exercises, the learning material, and the overall learning procedure."			

Text Representations of Math Tutorial Videos for Clustering, Retrieval, and Learning Gain Prediction

Pichayut Liamthong
Worcester Polytechnic Institute
pliamthong@wpi.edu

Jacob Whitehill
Worcester Polytechnic Institute
jrwhitehill@wpi.edu

ABSTRACT

With the goal of making vast collections of open educational resources (YouTube, Khan Academy, etc.) more useful to learners, we explored how automatically extractable text representations of math tutorial videos can help to categorize the videos, search through them for specific content, and predict the individual learning gains of students who watch them. In particular, (1) we devised novel text representations, based on the output of an automatic speech recognition system, that consider the frequency of different tokens (symbols, equations, etc.) as well as their proximity from each other in the transcript. Unsupervised learning experiments, conducted on 208 videos that explain 18 math problems about logarithms show that the clustering accuracy of our proposed methods reaches 85%, surpassing that of standard TF-IDF features (78% using log normalization). (2) In a video search setting, the proposed text features can significantly reduce the number of videos (up to 88% reduction on our dataset) and amount of video time (up to 82%) that users need to spend looking for desired content in large video collections. Finally, (3) in an experiment on Mechanical Turk with $n = 541$ participants who watched a randomly assigned tutorial video between a pretest & posttest, the text features and their multiplicative interactions with students' prior knowledge provide a statistically significant benefit to predicting individual learning gains.

Keywords: Open educational resources (OER), Crowdsourcing, Information Retrieval

1. INTRODUCTION

Consider a large repository (Khan Academy, edX, etc.) of open educational resources (OERs) such as tutorial videos, and a scenario in which the ultimate goal is to help learners to learn by recommending relevant and high-quality content that matches the students' needs. Knowing *what* the learner needs and providing the *right* content that suits them is crucial. We could estimate automatically the most beneficial content by analyzing their performance on prior examinations. However, a current challenge with contemporary OER repositories is that the content within each resource is typically poorly annotated, with tags that are too general, e.g., "algebra" or "linear equations" rather than "Simplify $\log_{10} 1000$.". Given the high labor and time involved in manual annotation, it is desirable to devise methods of *automatically* analyzing OER content and devising representations that can facilitate efficient search and categorization.

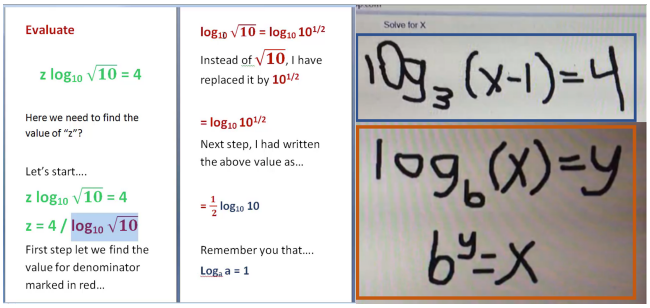


Figure 1: Example videos in our study. Right: Google's Speech-to-Text extracts the text "solve for x ok our problem is log base 3 of x minus 1 equals 4...".

While optimal character recognition and handwriting recognition are both mature fields, they are typically evaluated in much more constrained settings than math tutorials, in which math is mixed with natural language, and extraneous lines and other graphics can exist (see Figure 1). In full-fledged tutorial videos, this segmentation can be very challenging. Our research focuses instead on analyzing the speech transcript of the video (while ignoring other potential audio characteristics such as background noise, pitch, etc.). When a particular expression or equation is presented in a video, there is a high chance that the speaker will also say that expression/equation out-loud to the learners (Figure 1). Rather than manually transcribing the text from the video, we consider only fully automatic approaches based on automatic speech recognition (ASR; we used the Google Speech-to-Text API in our work, more detailed about the pilot test in Appendix B). Hence, the text representations we explore must contend with imperfect transcripts. We then assess the utility of the proposed representations for three tasks: (1) *cluster* the videos automatically into the specific math problems that they explain; (2) *search* through a library of videos for one that explains a particular math problem; and (3) *predict* the individual learning gains of students who

Pichayut Liamthong and Jacob Whitehill "Text Representations of Math Tutorial Videos for Clustering, Retrieval, and Learning Gain Prediction". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 661-666. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

watch the videos in a pretest/treatment/posttest paradigm. In these ways, we hope to make available to students the *right* content that is already available, but not easily findable, among large-scale OER repositories.

We conduct our investigation on a collection [14] of math tutorial videos about logarithms, and another dataset from YouTube on basic algebra. Our goal is not just to make coarse distinctions between videos about "algebra" versus "geometry", but rather fine-grained distinctions about specific math problems. Mirroring our goals from the previous paragraph, our research questions are the following: **RQ1**: How accurately can the devised text features cluster videos into *fine-grained* categories about the specific problem they are solving, and which aspects of these representations are most important? **RQ2**: By how much can we reduce the search time to find a relevant video? **RQ3**: Are the text features predictive of the individual learning gains of students who watch these videos in a pretest/posttest setting?

2. RELATED WORK

Text Representations: There are several prominent text representations used for language modeling: (1) Term frequency and Inverse document frequency (TF-IDF) [12]: TF-IDF features typically do not require training and are thus suitable for unsupervised settings. (2) Word embedding models [8, 7] based on neural networks trained using supervised learning. (3) Sentence-level models such as BERT [8] that capture higher semantics compared to word embeddings.

Video Categorization & Clustering: For categorizing video content automatically, much of the prior work has focused on other fields than math tutorial such as films, sport videos [2], [4], [11]. Most prior methods on video categorization focus on visual aspects such as frame transitions, object detection and segmentation. Some as them use the audio (e.g. [2]) such as the audio frequency and amplitude statistics. We are unaware of any previous research that clustered video content at the low-level tags of individual math problems.

Video Retrieval in OERs: There has been increasing interest in the task of video retrieval of OERs. Many works in have pursued combined feature representations with both textual and visual information [13, 15, 3]. Hürst [3] found that the lecture slides are more useful than the corrected transcriptions. In our work, while we focus solely on text representations, the features we devise could be easily combined with visual features.

Estimating the Effectiveness of OERs: For the task of estimating the effectiveness (e.g., associated learning gains) of viewing tutorial videos, researchers have pursued various approaches, including estimating their effectiveness through correlated measures such as engagement while watching the video [10, 6, 1]. For estimating the effectiveness of OERs in general, one can also use a combined experimental and reinforcement learning-based approach such as bandit algorithms [9]. While Rafferty et al. [9] suggested the potential use of *context* (for example, features of the OERs as well as of the students' prior knowledge) for predicting learning gains, they did not actually pursue that approach.

3. TEXT REPRESENTATIONS

In this paper we explore unsupervised representations of the transcripts of math tutorial videos. When designing the representations, we considered the following characteristics: (1) Similar content should involve similar tokens. A math video whose transcript consists of just "two plus three", for example, is unlikely to be similar to a video whose transcript is "four times x". (2) The most important tokens tend to recur within a video transcript. Conversely, tokens that are uttered only once are often less important or even be transcription errors. (3) The relative order of nearby tokens is important for deciphering the math content. For example, "four over two" and "two over four" are different fractions, but the difference is reflected only in the relative order of tokens, not in their frequencies. For characteristics (1) and (2) above, we created several variations of "1D" text representations that capture which tokens occur more frequently in each video. With the additional characteristic (3), we also explored "2D" text representations that can capture the relative order within a fixed radius from token i w.r.t. token j for each (i, j) pair. We note that extracting the precise mathematical expression from the transcript is inherently ambiguous. For example, the two distinct expressions 2^{x+2} and $2^x + 2$ would likely both be spoken as "two to the x plus two". Fortunately, our objective is not to capture the math content perfectly, but to capture enough of it to enable effective clustering, search, and prediction of learning gains. Below we describe different kinds of unsupervised text representations that vary in terms of token type, order dependency, and summarization method.

3.1 Token Types

3.1.1 Individual Token

As our simplest representation, we call each word (separated by space) a *token*, and then we count the number of math-related tokens, defined as: (1) numbers (digit-only), (2) operations (e.g. $+, -, \times$), or (3) variables (an alphabet). For the operations, we map synonyms to the same token, e.g., 'plus' to '+', 'to the [power]' to '^'. Additionally, we add the words corresponding to each digit 0 to 9 (i.e. 'zero', ..., 'nine') as math-related tokens. For variables, we used a restricted alphabet consisting of $\{b, c, n, m, w, x, y, z\}$ (we omitted 'a' since it is also a common English word).

3.1.2 Expression Token

To infer which math problem in video, it might be useful to extract the entire expression.For example, "2 plus 3" could be considered as one token "2+3" not '2', '+', and '3'. Specifically: (1) We mark all tokens in the transcript as either math-related or non-math-related. Tokens that are labeled as math-related are literals (LIT) and operators (OP) such as plus (+), $\sqrt{}$, etc. (2) For each contiguous sequence of math-related tokens, we read the tokens one-by-one and concatenate them into one expression according to the rule: starting with LIT followed by OP, LIT, ... (alternately).

3.2 Token Count Vector

Given the sequence of tokens in each video, we then compute either a 1D vector or 2D matrix of frequency statistics (which are finally summarized as described in Section 3.3). In the subsections below we let $\mathcal{T}$ be the set of all tokens that appear in any of the videos.

1D (No Order Dependencies): The count vector of each video contains $|\mathcal{T}|$ components, each of which records the frequency of token occurs in video.

2D (First-Order Dependencies): With the goal of encoding the *relative order* of tokens, we computed a 2D *matrix* M, of size $|\mathcal{T}| \times |\mathcal{T}|$, such that M_{ij} is the number of times that token i appears before token j in the transcript. In this approach, we introduced a "radius" parameter k to limit the distance of token pairs (i, j) that need to be considered. For example, if $k = 4$, all token pairs (i, j) such that the distance between i and j is ≤ 4 will be counted, otherwise, ignored.

3.3 Token Summarization Methods

Given the token count vector computed in Section 3.2, we then summarize each count x using a summarization function f. We considered the following functions: (1) **Raw Frequencies**: We let $f(x) = x$. (2) **Binarized Frequencies**: Binarizing the counts x might be less susceptible to noise; hence, we tried setting: $f(x) = 1$ if $x \geq 1$ and $f(x) = 0$ if $x = 0$. (3) **Weighted Frequencies**: It might be beneficial to weight down tokens which appears once because it might be noise from the extraction process; important tokens should be mentioned multiple times in video; token found only once (we call it $t_{=1}$) are either insignificant or incorrectly extracted. Instead of removing $t_{=1}$, we introduced the parameter r to downweight $t_{=1}$. In this case, instead of having the raw frequencies, We fixed the weight of $t_{>1}$ (appear *more than* once) as 1; however, we downweight $t_{=1}$ by r. We thus let $f(x) = 1/r$ if $x = 1$, $f(x) = 0$ if $x = 0$, and $f(x) = 1$ if $x > 1$. Note that $r = 1$ is equivalent to Binarized Frequencies.

4. DATASET

We applied the text representations to two sets of tutorial videos: (1) *Logarithms* and (2) *Algebra*, see Appendix A.

5. CLUSTERING

Given the different feature types, we test whether they serve as an effective basis for clustering the videos. In this section, as ground-truth cluster labels, we took the math problem (there were $K = 18$ unique problems in total) that each video explained as its label. Note, however, that we could also cluster the videos by the *category* of problems that they explain (see Section 4); we do so in Section 7.

Methods: For each of the different text representations, we applied K-means clustering to group the videos into $K = 18$ clusters, followed by the Hungarian algorithm [5] to optimally match the estimated cluster to the ground-truth indices. Since K-means converges to different local minima depending on the random initialization, we executed the algorithm 512 times and then reported the average of accuracy for the clustering with lowest sum of squared distance.

Results: Table 1 shows the clustering accuracy results. All three methods yield accuracies that are much greater than the *random* baseline, which achieved only 18.27% accuracy.

Weighted Frequencies: We tried multiple values of r ($r = 1$ is equivalent to the Binarized Token Counts). We also added $r = 0.5, 0.25$; this contrasts with our intuition for when it weights $t_{=1}$ more; we added this as a sanity check that the accuracy should be getting worse. Table 1 shows that the weighted frequencies increase the accuracy significantly up by 10% on average. $r = 2$ performs the best among $r = 2, 4, 8$. As r gets larger, we see a slight decrease in accuracy.

1D vs. 2D: Table 1 (right) shows clustering accuracy with the 2D approach. For the radius $k = 2$ on the Expression token (and using weighted frequencies with $r = 2$), the accuracy increases around 2% compared to with 1D. However, we can see lots of variance in the accuracy over the different k, and hence the advantage may not be statistically reliable.

Comparison to TF-IDF: Our token summarization methods can be seen as variations of TF-IDF, where only the TF term $f(x)$ is used; in other words, we used a constant 1 for the IDF term. (We experimented with several IDF functions but found that they all worked worse than just 1.) The weighted frequency scheme we tried can be seen as a coarse (piecewise-constant) approximation to the (smooth) log function commonly used as the TF function in TF-IDF. Using TF-IDF (with log for TF and 1 for IDF) and Expression Tokens, the clustering achieved 78.67% for the Expression Token (down about 5% from our *weighted* frequency method). For the Individual Tokens, it performed similarly in accuracy compared to the weighted frequency methods. (See the "log" column in Table 1.) In summary, the results provide some evidence that our text representations may yield a worthwhile accuracy advantage over TF-IDF.

6. SEARCHING

Here we explore whether the proposed text representations could be used to create a simple search engine to reduce the amount of *video time* they would need to watch. Using the text representations, we can build a simple search engine as follows: (1) From each video i in a collection $\mathcal{S}$, we transcribe its speech into text (using Google ASR) and then extract its text representation v_i. Then, (2) for any search query (e.g., "Simplify: $\log_4 16$"), we likewise extract its text representation q using any of the methods presented in Section 3. Finally, (3) we rank all the videos in $\mathcal{S}$ by the *cosine similarity* between v_i and q.

Experiments: Here we consider a general setting in which *multiple* math problems may be explained in a *single* video. A search engine that can pinpoint which *segment* of a video explains the solution could save the user significant time compared to watching the whole video. For this setting, there is a trade-off between granularity and accuracy: the search engine may be more accurate if the segment length is longer, but the user can save more time if the segment returned to them by the search engine is shorter. Hence, we introduced a segment length parameter, L. We divided each video into multiple segments of length L. Each segment has its own (sub-)transcript and its own problem that it explains. Hence, we treat each segment as its own "video". Our goal is to find *any* segment in the video that explains the problem in the user's query q. As a baseline, we used a simulation (averaged over 20 runs) to estimate the sum of the segment lengths (in seconds) that a user would have to watch before finding a relevant segment.

Results on the Algebra dataset: We analyzed the 234 videos of the Algebra dataset that contain multiple problems; in

Token Types	1D								2D					
	Raw	Weighted: r						log	Radius k (for Weighted: $r = 2$)					
		0.25	0.5	1	2	4	∞		1	2	4	8	16	∞
Individual	54.33	25.48	41.35	67.31	72.11	68.75	64.90	73.56	64.90	63.46	59.13	57.69	62.02	49.04
Expression	50.48	20.67	44.71	70.19	**83.65**	83.17	80.29	78.67	83.65	**85.58**	75.96	71.64	73.56	51.44

Table 1: Clustering accuracy on the logarithm videos for 1D text representations with different token types and summarization methods, and the clustering accuracy for 2D representations and different token types (all with weighted summarization: $r = 2$).

total, these videos explain 300 algebra problems. We varied the segment length L over the set {15s, 30s, 1m, 2m, 4m} (see Figure C.1 in Appendix). The results shows that the best text representations were 2D Binarized Individual Token ($k = 8$). In particular, the 2D representations showed an advantage (compare the pairs of {blue, pink}'s *solid* and *dashed* lines). We found that radius $k = 8$ for 2D Representation preforms best across each method. For the Interval Length, the percent decrease, at $L \in \{30s, 1m\}$, in watch time is highest (i.e., the most helpful, see Figure C.1 in Appendix). As L continues to grow, the results go down and at $L = 15s$, the performance drops. This exemplifies the trade-off between segment length and available information.

Results on the Logarithm dataset: In this dataset, each video contains *one* log problem. For each of 18 logarithm problems, we search for *any* of the videos that solve that particular problem. Comparing the results with *random* baseline, the results show the same trend as for the Algebra dataset: The 2D Representation gives the best results. We found, for instance, Binarized Individual Token yields the results of 89.96%, and 93.19% for 1D and 2D ($k = 8$), respectively. The same holds true for Weighted Expression Token ($r = 2$) with the results of 91.25% and 93.20%. For the 1D approach, the best representation was TF-IDF (with log for TF and identity for IDF); the reduction was slightly lower (92.85%).

7. LEARNING GAIN PREDICTION

In this section, we investigate whether the text representation can be used to predict the learning gain of students who watch the videos as an educational intervention. The high-level idea is that the effectiveness of each tutorial video can be estimated by the *interaction* of the *content* within the video and the student's prior knowledge. In contrast to some prior work that predicted the *average* learning gains of a video over many students, here we tackle the arguably harder problem of predicting *individual learning gains* of each student, measured as the difference in test scores on the curriculum before and after watching the video.

The Logarithms dataset (Section 4) contains pretest/posttest scores of students who received a tutorial videos as an intervention. Hence, we use each participant's pretest score and the text representation of the video they watched as predictors to estimate their learning gains (posttest minus pretest score). Rather than use the text representation as a feature vector itself, we instead use the *category label* assigned to the problem (Section 5) by the clustering algorithm as a 0-1 indicator variable with an associated model coefficient; hence, our models can find interactions between a student's prior knowledge and the topic in the video they received.

7.1 Prediction Models

We considered both linear models with mixed effects, as well as deep non-linear models based on neural networks, but we found that the latter overfit too easily and gave unstable results; hence, we present only the linear models. Let $p_{ij}, j = 1, 2, 3$, be student i's prior knowledge (pretest score) within the 3 problem categories (j) on logarithms. Let $c_{ij}, j = 1, 2, 3$, be 0-1 indicator variables that reflect whether student i's assigned video belongs to each category j. (Note that each video is assigned to exactly one of the three categories.) We can compute c_{ij} using either (a) Manually Labeled Categories (MLC) from human annotators, or (b) Automatically Labeled Categories (ALC) from the text representations and clustering algorithm (Section 5).

Prediction Model: We constructed a model that considers multiplicative interactions between the student's prior knowledge p_{ij} in each problem category and the cluster label c_{ij} of the student's assigned video:
$y_i = \sum_{j=1}^{3} (w_j p_{ij} + v_j c_{ij} + u_j (p_{ij} \times c_{ij})) + \epsilon_i$. Importantly, this model contains multiplicative interaction terms $p_{ij} \times c_{ij}$.

Results: We found that the interaction $p_{ij} \times c_{ij}$ using MLC has a statistically significant effect on the learning gain ($F_{11,\,582} = 5.839, p = 5.11e-09$), and so does this interaction using ALC ($F_{11,\,582} = 6.425, p = 4.125e-10$). The RMSE is 0.464, which is slightly better (about 3.1% relative decrease) compared to prediction model 1. Specifically, we found that, for example, u_3 is *negative* and statistically significant ($p = 0.0005$) in the ALC model. The negativity of u_3 means that, if $p_{i3} \times c_{i3}$ is low, then the learning gain is high (and vice versa). In turn, $p_{i3} \times c_{i3}$ is low either because (1) p_{i3} is low and $c_{i3} = 1$, i.e., an individual knows little about topic 3 and receives a tutorial about topic 3, yielding high learning gain; or (2) p_{i3} is high and $c_{i3} = 0$, i.e., an individual already understands topic 3 and receives on another (more helpful) topic, yielding high learning gain. Both the MLC and ALC interactions were stat. sig., suggesting that the text representations can group videos in ways that predict individual learning gains.

8. CONCLUSION AND FUTURE WORK

We have devised novel text representations to represent the content of math tutorial videos. On a dataset of hundreds of math videos and hundreds of students who watched them, we showed that the representation can be used to (1) accurately (around 85%) cluster the videos into the math problems they solve (RQ1); (2) search for specific video content in a large repository of videos, thereby saving the user considerable (up to 88%) search time (RQ2); and (3) predict individual learning gains, in conjunction with features of the students' prior knowledge, with stat. significance (RQ3).

9. REFERENCES

[1] S. Bulathwela, M. Pérez-Ortiz, A. Lipani, E. Yilmaz, and J. Shawe-Taylor. Predicting engagement in video lectures. *arXiv preprint arXiv:2006.00592*, 2020.

[2] S. Fischer, R. Lienhart, and W. Effelsberg. Automatic recognition of film genres. *Technical reports*, 95, 1995.

[3] W. Hürst, T. Kreuzer, and M. Wiesenhütter. A qualitative study towards using large vocabulary automatic speech recognition to index recorded presentations for search and access over the web. In *ICWI*, pages 135–143. Citeseer, 2002.

[4] V. Kobla, D. DeMenthon, and D. Doermann. Detection of slow-motion replay sequences for identifying sports videos. In *1999 IEEE Third Workshop on Multimedia Signal Processing (Cat. No. 99TH8451)*, pages 135–140. IEEE, 1999.

[5] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[6] A. S. Lan, C. G. Brinton, T.-Y. Yang, and M. Chiang. Behavior-based latent variable model for learner engagement. *International Educational Data Mining Society*, 2017.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[8] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[9] A. N. Rafferty, H. Ying, and J. J. Williams. Bandit assignment for educational experiments: Benefits to students versus statistical power. In *International Conference on Artificial Intelligence in Education*, pages 286–290. Springer, 2018.

[10] A. Ramesh, D. Goldwasser, B. Huang, H. Daumé III, and L. Getoor. Learning latent engagement patterns of students in online courses. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1272–1278, 2014.

[11] E. Sahouria and A. Zakhor. Content analysis of video using principal components. *IEEE transactions on circuits and systems for video technology*, 9(8):1290–1298, 1999.

[12] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[13] F. Wang, C.-W. Ngo, and T.-C. Pong. Structuring low-quality videotaped lectures for cross-reference browsing by video text analysis. *Pattern Recognition*, 41(10):3257–3269, 2008.

[14] J. Whitehill and M. Seltzer. A crowdsourcing approach to collecting tutorial videos–toward personalized learning-at-scale. In *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*, pages 157–160, 2017.

[15] H. Yang and C. Meinel. Content based lecture video retrieval using speech and video text information. *IEEE transactions on learning technologies*, 7(2):142–154, 2014.

APPENDIX

A. DATASET

Here we described each dataset we use in the experiment in more detail.

Logarithms: This is the dataset collected by Whitehill & Seltzer [14], which contains both a repository of 208 math tutorial videos about logarithms. Most videos are between 1-3 minutes long. In total the collection spans 18 logarithm problems, with 9 to 17 videos per problem. Relevant only to Section 7, the dataset also contains students' pretest and posttest scores of 541 participants from Amazon Mechanical Turk who watched the videos. There are 226 males, 207 females, and 108 of undefined, with the average age of 33.71 $\pm$9.84. Specifically, each participants were asked to answer 19 logarithm pretest problems, which was classified into 3 main categories: (1) the logarithmic term without variables e.g. $\log_9 1$, (2) the logarithmic term with variables e.g. $\log_w \frac{1}{w}$, and (3) the logarithmic equation e.g. solve for x where $x \log_4 16 = 3$ (category 1, 2 and 3 contain 102, 61, and 45 videos, respectively). Then, they were assigned to *one* random video among 208 logarithm tutorial videos, and were asked to complete a posttest (same level of difficulty as the pretest but slightly different problems).

Algebra: For the search task, we collected another dataset, containing 234 algebra math tutorials on Youtube As of 234 videos, 213 of them contains *one* math problem and 21 of them contains *multiple* math problems (total of 87 math equations); total of 300 expressions on entire dataset. We manually annotated which equation (e.g. $2x^2 - 2x - 12 = 0$, $x + 7 = 10$) each video explains. For videos with multiple math problems, we marked the start end time of each.

B. SPEECH-TO-TEXT TRANSCRIPTION

All the feature types we explore are based on obtaining an *approximate* transcript of the video from an ASR. In particular, we use Google Speech-to-Text API. As a pilot test of its accuracy on the OERs in our dataset, we manually annotated 10 videos (in total of 3044 words in the ground-truth transcripts). Google's API achieved a word error rate (WER) of 5%, which intuitively seemed sufficient, which intuitively seemed sufficient. An example of extracted speech is shown in Figure 1 (caption). After obtaining the transcript for each video in our collection, we then tokenized it and summarized the token frequencies.

C. ADDITIONAL FIGURES

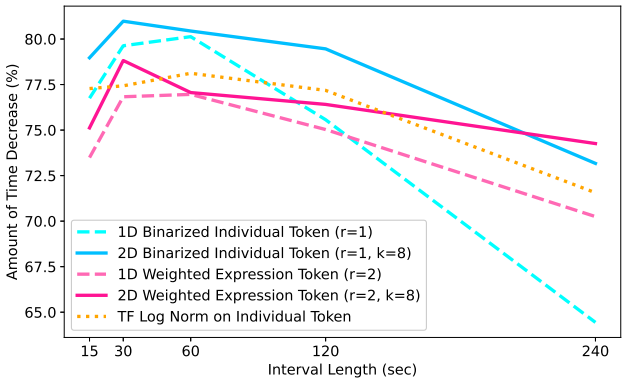


Figure C.1: The decrease in time needed to find specific math content in a set of math tutorial videos. Each line shows a different text representation over different segment lengths.

Predicting Young Students' Self-Evaluation Deficits Through Their Activity Traces

Thomas Sergent
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
Lalilo, Paris, France
thomas.sergent@lip6.fr

Morgane Daniel
Lalilo, Paris, France
morgane@lalilo.com

François Bouchet, Thibault Carron
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
{francois.bouchet, thibault.carron}@lip6.fr

ABSTRACT

Self-evaluation is a key self-regulatory process that can already be mastered by young children. In order to assess self-evaluation skills of children, we introduced a random prompt asked randomly after 1 out of 15 exercises into a literacy web-application for primary school student, in order to evaluate the perceived difficulty [Too easy, Good, Too difficult] of the exercise they just solved. Comparing students' actual performance with their responses to this prompt can provide information about their ability to self-evaluate, and thus detect students who could improve their self-evaluation skills. We collected more than 1,000,000 responses from 300,000 students and used these data as well as performance data on each question of each exercise to predict a student's response to the next prompt, thereby estimating how likely they are to having a self-evaluation deficit. The results show (a) that a student's past responses to self-evaluation statements impacts the quality of future predictions (b) that the impact of past responses - vs their current performance - is greater when the student has low capacity for self-evaluation (c) that including older student data (answers from several sessions ago) helps in improving the accuracy of the prediction. These results pave the way (1) for adaptive polling by identifying when the model is unreliable, giving them the statement then instead of randomly, (2) for adaptive feedback, by knowing the students the most likely to show a deficit, to provide remediation.

Keywords

Self-Regulated Learning, Primary school, Self-evaluation, Prediction, Remediation, Adaptive polling

1. INTRODUCTION

Improving children's self-regulated learning (SRL) skills is a key component of their academic performance, as self-regulated students generally know better "how to learn", which can have a positive impact in all disciplines [22]. A key SRL process is self-evaluation [17], which is a skill already developed in children as young as 5 years old [19]. It is therefore a particularly interesting SRL aspect to target when working with young children. The most reliable way to assess SRL deficits is through direct questions to the students [1], but constant prompting can lead to an overall degraded perception of the learning environment [3] and it is therefore critical to limit prompting to the minimum. Hence we are interested here in trying to predict students' answers to assessment on perceived difficulty which are used to assess students' tendencies to overevaluate or underevaluate. The second aspect we investigate is relative to the features that are the most relevant for this prediction.

2. RELATED WORK

The EDM community has recognized early on the interest of studying SRL through data mining [21], and previous works have been more particularly interested in detecting SRL behaviors from traces [5], discussion forums [9, 8] or proxy behaviors such as gaming the system [4], explaining such behaviors with sequence mining [20, 2], mixture models (for procrastination, a proxy of SRL) [13] or coherence analysis [18]. Other works have focused on analyzing the differences in use of SRL strategies [6], providing feedback to encourage them [7] or predicting their use [15]. However, as far as the authors are aware, no previous work has specifically attempted to predict how a student would answer to a question aiming to measure a SRL deficit (self-evaluation or any other), and none of the aforementioned work focused on young children (5-7 years old). It is worth noting that although young children's abilities to use SRL strategies may be more limited than in teenagers, they seem to have comparable monitoring skills [16]. Indeed, recent work on a dashboard supporting SRL in a mathematics software program for 9-10 years old (only slightly older than our targeted students) showed a significant improvement in SRL skills for students in the dashboard group compared to those without the dashboard [12].

3. SELF-EVALUATION ASSESSMENT

3.1 Context

Lalilo is one of the many web applications used by teachers in the classroom to help them implement a differentiated pedagogy. At the beginning of 2021, it is used by 40,000 English and French speaking kindergarten and elementary classes every week to strengthen literacy through series of exercises adapted to the students' level, while providing the

Thomas Sergent, Morgane Daniel, François Bouchet and Thibault Carron "Predicting Young Students' Self-Regulated Learning Deficits Through Their Activity Traces". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 667-671. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

teacher with a dashboard to evaluate the students' activities and progress. It is therefore a relevant testing ground for evaluating and then trying to correct some students' SRL deficits. A typical session lasts 20 minutes (on average) with the student performing around 15 short exercises with 3 to 7 questions each. Student activities (e.g. logging in, time spent on an question/exercise, mistakes) are traced and we will focus on students' answers to an exercise, thus we will call **trace** only the answers to this set of questions of the same type within an exercise.

3.2 Data collection

To assess some aspects of students' SRL skills, we introduced a random prompt is once every fifteen exercises when a student finishes an exercise (i.e. on average once per typical learning session). This prompt includes a **perceived difficulty** statement asking the student *"How difficult was this exercise for you?"* with 3 possible answers: "Too hard", "Just-right", "Too easy"). Comparing the answer to the perceived difficulty statement with the real performance aims at measuring the **self-evaluation** ability of the students, i.e. their ability to correctly estimate the difficulty of the questions they just answered. Before introducing the assessments, we checked qualitatively in a classroom using Lalilo that statements were understood by 1st grade students (details not presented here). We collected traces from Kindergarten, 1st grade and 2nd grade classes based in France, Canada and USA learning in French (FR) or English (EN) between January 18 and February 24, 2021 on the Lalilo platform. We kept only the traces for which students had answered to a prompt and further on we call trace the answers to the exercise with the associated answers to the prompt.

4. METHODS

4.1 Dataset

Given the history of a student answers to the perceived difficulty prompt and their performance on the current exercise, we want to predict which answer is the most likely to be given by the student to the next prompt (and thus extrapolate their self-evaluation skills). If the student's performance - which will be defined in the Feature engineering subsection - was "excellent" (resp. "poor") and they answered "Too hard" (resp "Too easy"), they were considered as having an underevaluation (resp. overevaluation) deficit. We filtered out students who had strictly less than 8 traces with answers to prompts so that our model would not overfit on results of students with very few answers, as students with very few answers were overrepresented in our initial dataset. We finally had 424,173 traces with an answer to the perceived difficulty statement from 34,083 students having on average 12 traces with self-evaluation answers (SD = 5.93).

4.2 Feature engineering

We engineered several new features that could have a predicting power in our results.

Basic performance feature. In addition to the trace and student IDs, used for filtering but not for prediction, we extracted for each trace the *answer correctness list*, a boolean vector of a length of 3 to 7 (number of questions per exercise).

Enriched performance features. From the answer correctness list, we extracted 5 additional features: the *good answer count* (i.e. the number of 1s in the vector - the higher the value, the better the student may feel they have succeeded), the *total answer count* (i.e. the length of the vector), *the success rate* (i.e. the ratio between the good answer count and the total answer count) and the *second half success rate* (i.e. the success rate on the last half of a trace - a student with self-evaluation deficit may suffer from a recency bias, influencing positively [resp. negatively] their perception of their performance when answering correctly [resp. incorrectly] to the last questions of the exercise).

Exercise features. We hypothesize that self-evaluation deficits are not uniform across one's knowledge. In particular, a student's deficit may be stronger in some types of exercises or on exercises about a given topic. To assess this impact, we added 5 features that relate to the exercise finished just before the two difficulty statements: *exercise template* (for example a multiple choice question or a word composition exercise), *lesson index* (there are around 1,000 lessons in Lalilo - although they are not entirely linear, the higher the value of this feature, the more advanced the content is; when working on English (resp. French) data, the lesson index FR (resp. EN) is empty), *lesson type* (lessons are organized in a tree structure - lesson type represents the first level category), *lesson subject* (lesson subject represents the second level category in the tree), *language* (English or French).

Previous feedback modalities. In order to help students' in their performance assessment, they are randomly given an audio feedback (such as "In the last exercise, you found 3 correct answers of 5 questions") and/or a visual gauge of as many green ticks and red crosses as they had good and bad answers in the previous exercise. Even when these synthesis exercise-level performance indicators are not there, the students always have an immediate question-level true/false feedback. We have previously shown the positive impact of these two indicators in correcting some self-regulation issues, and we therefore hypothesize that they need to be taken into account when predicting how the student will assess the difficulty. Hence we added two binary features, *gauge* and *audio* which indicate whether these performance feedback were given before the two difficulty statements. A feedback is also provided after answering to the prompt, when students display a self-evaluation deficit (as defined in 3.2), encouraging them to be more confident or warning them to be more careful; we therefore encode this as a third binary feature, *remediation*, indicating whether the student received a feedback the last time the difficulty was assessed.

Self-evaluation deficit lag features. Self-evaluation deficits are expected to be a recurring phenomena in students' answers, i.e. a student who has under/overevaluated themselves a few times is likely to under/overevaluate themselves again in the future. Hence we added 3 lag features for the last 3 perceived difficulty assessments. Moreover, since it is possible that the last 3 assessments were not allowing the student to exhibit a deficit (e.g. a student cannot appear to be overevaluating if their performance is at 100% on the last 3 exercises where they were asked to assess the difficulty), we also added 3 lag features for the last 3 perceived difficulty assessments where the student's performance was equal to the performance on the current exercise. Performance is a categorical value which is worth "poor" if the success rate is below 34%, "excellent" if the success rate is at 100% and "medium" otherwise.

Overall previous self-evaluation deficit. If students are stable over time in their assessment, we expect that taking into account the whole history would have a positive impact on

the prediction. We therefore introduced 5 additional features: *self-evaluation answer rank* (the number of times the student has been asked a self-assessment), *number of "Too easy" (resp. "Too hard") answers* (the number of times the student previously answered "Too easy" (resp. "Too hard") to previous assessments), and *"Too easy (resp. Too hard) answers ratio"* (the ratio between the two previous features). Additionally, similarly to what was done for the lag features, we also considered the 5 equivalent features exclusively on previous assessment given after an exercise with a similar performance level.

4.3 Algorithms

We used Catboost [14] and LightGBM [10] to perform the predictions, two gradient boosting algorithms based on Decision Trees whose main assets are: (a) their ability to natively deal with categorical features and (b) their explainability, allowing to study feature importance in each prediction with SHapley Additive exPlanations (also called SHAP values) [11]. They also have recently won Kaggle competitions on a variety of datasets. We used MultiClass as a loss function, set the number of iterations at 200 and kept the other hyperparameters at their default values. As their results were very similar for the global prediction task (cf. Table 1), we used CatBoost model only for the other tasks.

4.4 Analyzing features importance

We first measured the improvements allowed by feature engineering to the global prediction scores using 5-fold cross validation and stratifying by student so that no student traces are both in training and testing folds. For all feature importance measures subsequently described, we created a training and a testing set - also stratified by student - and measured the feature importance in the testing set. We studied the importance of various features in our model using the SHAP package [11]. We also compared the importance of features across the three classes so as to highlight the features that have the most impact on each class specifically.

If students showed a given deficit regularly - we defined a threshold at 50% of "underevaluation" (resp. "overevaluation") over the traces with 100% (resp. below 34%) success rate - we tagged the student as having an "underevaluation (resp. overevaluation) deficit". We then trained our algorithms on a dataset with students tagged as having a deficit and on a dataset with students tagged as having no deficit. Our hypothesis was that feature importance would vary between these two models: the predictions were expected to depend more on past answers than current performance for students tagged with deficits compared to the other students.

Finally, we measured the evolution of the performance of the model depending on the self-evaluation answer rank that is predicted. To do so, we analyze the evolution of Cohen's Kappa coefficient, measuring the quality of our prediction of the perceived difficulty, on a cohort of students of the testing set having answered a given amount of prompts. We computed this coefficient for each self-evaluation answer rank and expected the quality of the prediction to improve as students would be better and better characterized by their features.

5. RESULTS AND DISCUSSION

5.1 Global features importance analysis

Firstly, both Catboost and LightGBM allow to predict with a reasonable overall performance the students' perceived difficulty (cf. right part of Table 1). Secondly we see that all features additions improve the model except the previous feedback ones. Specifically, adding features describing what a student did in the past improves the predictions significantly.

Table 2 ranks the top 10 features with their mean absolute SHAP values. Interestingly, the success rate of the student and the success rate on the second half of the correctness list are only the 5^{th} and 8^{th} ranked features. It means that past information about the previous answers of students to self-evaluation prompts influences more the prediction than their current performance, although they are being asked "How difficult was **this** exercise for you?". Specifically, the last three answers to the perceived difficulty question bear a significant weight in our model's predictions, as well as the global "Too easy" and "Too hard" ratios. As expected, the "Too easy" ratio has a huge importance for the "Too easy" class as has the "Too hard" ratio for the "Too hard" class and both ratio are highly ranked for the "Just-right" class. Indeed, we did not input the "Just-right" ratio as the model can learn it from the combination of "Too hard" and "Too easy" ratio. We can note that the success rate feature is mainly important for the "Too easy" and "Too hard" class, which is logical as an excellent (resp. poor) performance is not likely to lead to a "Too hard" (resp. "Too easy") answer. We can also see on Figure 1, as the "Too hard" ratio is equal to 0, it drives the prediction score of the "Too hard" class downwards while it drives the prediction score of the "Just-right" class upwards. Furthermore, the 3 last answers to the perceived difficulty statement of this student were "Just-right", "Too easy", "Too easy" and their success rate on this trace was 100%; the predictions of the "Too easy" class are therefore also driven upward by these features.

5.2 Features rank from self-evaluation deficits

Figure 2 shows the feature importance rankings for students detected as having or not self-evaluation deficits. Students with deficits consistently choose how to answer to the prompt more based on past answers (in particular the "Too easy/hard" ratios), as opposed to students with no deficits who rely more on the success rate to this exercise, as one should. These results are in line with our hypotheses.

5.3 Predicting power based on answer rank

Figure 3 shows the kappa evolution depending on the number of past self-evaluation assessments. The kappa for the first answer is around 0.13, then quickly climbs around 0.4 for the next four traces; and finally slowly increases until plateauing around 0.6. The kappa of 0.13 for the first answer is consistent with Table 1: at the beginning, Student features are empty and the model can only rely on Trace features. With Trace features only, the model reached a Kappa of 0.1084 which coincides with the kappa value of 0.13 in Figure 3. We can then deduce that the model is more and more able to predict answers to the perceived difficulty statement.

Table 1: Prediction metrics after 200 iterations of the Catboost and LightGBM algorithms depending on the features used. We measure mean and standard deviation (in parenthesis) on 5-fold cross validation.

	Trace features			Student features			
Algorithm	Enriched perf.	Exercise	Feedback	Lag	Overall prev. self-eval.	Accuracy	Kappa
CatBoost	Yes					0.4662 (0.0006)	0.0866 (0.0030)
CatBoost	Yes	Yes				0.4737 (0.0018)	0.1084 (0.0022)
CatBoost	Yes	Yes	Yes			0.4736 (0.0018)	0.1081 (0.0026)
CatBoost	Yes	Yes	Yes	Yes		0.6676 (0.0034)	0.4522 (0.0053)
CatBoost	Yes	Yes	Yes	Yes	Yes	**0.6701** (0.0028)	**0.4575** (0.0042)
LightGBM	Yes	Yes	Yes	Yes	Yes	**0.6706** (0.0034)	**0.4565** (0.0032)

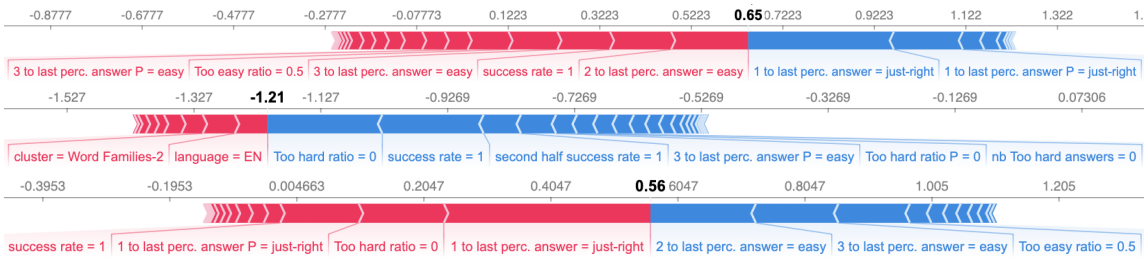


Figure 1: Feature impact in the prediction of each class for a randomly chosen trace in the testing pool. Top: "Too easy" class, middle: "Too hard" class, bottom: "Just-right" class.

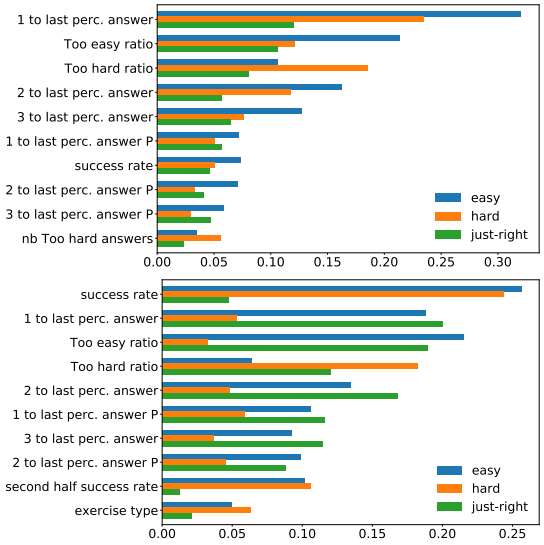


Figure 2: Top 10 features impact for class prediction of students detected as having (top) or not (bottom) a self-evaluation deficit

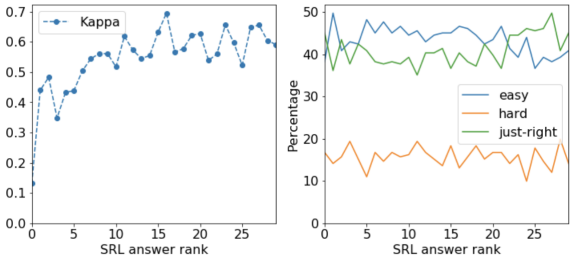


Figure 3: Kappa value and total number of traces of each class in the testing group, based on the self-evaluation answer rank

Table 2: Average feature importance rank by class, sorted by total SHAP value (top 5 in bold)

Features	Too easy	Just-right	Too hard
"Too easy" ratio	**1**	**1**	14
1 to last perc. answer	**2**	**2**	**3**
"Too hard" ratio	6	4	**1**
2 to last perc. answer	**3**	**3**	9
success rate	**4**	9	**2**
3 to last perc. answer	**5**	**5**	10
1 to last perc. answer P	7	6	12
second half success rate	8	19	**4**
3 to last perc. answer P	9	7	11
exercise type	10	15	**5**

6. CONCLUSION AND FUTURE WORKS

Using a large volume of trace data from primary school students, we leveraged students' past data to significantly improve the prediction of the answers to future self-evaluation prompts. The results also indicate that the more data we have about a student, the better our predictions are. Using feature engineering, we ranked features by the additional predicting power they provide, and found results consistent with SRL theories (in particular that prediction of answers for well-regulated students depends mostly on their success rate). This paves the way for adaptive polling (as opposed to the current random one), prompting only students likely to display a self-evaluation deficit, allowing us to better target remediation. The main limit of the current work is the specificity of the context: it would be particularly interesting to study the main features used in another context with a different type of students. We are also targeting one of many existing SRL deficits, and expanding research on predicting other deficits to encourage the training of multiple SRL skills seems important as well. Future works also include further feature engineering to refine what features may have more impact than the current ones.

7. REFERENCES

[1] L. Barnard, W. Y. Lan, Y. M. To, V. O. Paton, and S.-L. Lai. Measuring self-regulation in online and blended learning environments. *The Internet and Higher Education*, 12(1):1–6, Jan. 2009.

[2] F. Bouchet, J. M. Harley, and R. Azevedo. Impact of Different Pedagogical Agents' Adaptive Self-Regulated Prompting Strategies on Learning with MetaTutor. In *Proc. of the 16th Conf. on Artificial Intelligence in Education (AIED 2013)*, volume 7926 of *Lecture Notes in Computer Science*, pages 815–819, Memphis, TN, July 2013. Springer Berlin Heidelberg.

[3] F. Bouchet, J. M. Harley, and R. Azevedo. Evaluating Adaptive Pedagogical Agents' Prompting Strategies Effect on Students' Emotions. In *Intelligent Tutoring Systems: 14th Int. Conf.*, volume 10858 of *LNCS*, pages 33–43, Montreal, QC, Canada, June 2018. Springer.

[4] S. Dang and K. Koedinger. Exploring the Link between Motivations and Gaming. In *Proc. of the 12th Int. Conf. of Educ. Data Mining*, pages 276–281, 2019.

[5] N. Diana, M. Eagle, J. C. Stamper, and K. R. Koedinger. Extracting Measures of Active Learning and Student Self-Regulated Learning Strategies from MOOC Data. In *Proc. of the 9th Int. Conf. on Educ. Data Mining*, pages 583–584, Raleigh, NC, USA, 2016.

[6] E. Farhana, T. Rutherford, and C. F. Lynch. Investigating Relations between Self-Regulated Reading Behaviors and Science Question Difficulty. In *Proc. of the 13th Int. Conf. on Educ. Data Mining*, pages 395–402, Ifrane, Marocco, 2020.

[7] J. Feild. Improving Student Performance Using Nudge Analytics. In *Proc. of the 8th Int. Conf. on Educ. Data Mining*, Madrid, Spain, 2015.

[8] F. Harrak, F. Bouchet, V. Luengo, and R. Bachelet. Automatic Identification of Questions in MOOC Forums and Association with Self-Regulated Learning. In *Proc. of the 12th Int. Conf. on Educ. Data Mining*, pages 564–567, Montréal, Canada, July 2019.

[9] E. Huang, H. Valdiviejas, and N. Bosch. I'm Sure! Automatic Detection of Metacognition in Online Course Discussion Forums. In *Proc. of the 8th Int. Conf. on Affective Computing and Intelligent Interaction (ACII)*, pages 1–7, Sept. 2019.

[10] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, 30, 2017.

[11] S. M. Lundberg and S.-I. Lee. A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, 30, 2017.

[12] I. Molenaar, A. Horvers, R. Dijkstra, and R. S. Baker. Personalized visualizations to promote young learners' SRL: the learning path app. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 330–339, 2020.

[13] J. Park, R. Yu, F. Rodriguez, R. Baker, P. Smyth, and M. Warschauer. Understanding Student Procrastination via Mixture Models. In *Proc. of the 11th Int. Conf. of Educ. Data Mining*, pages 187–197, Buffalo, NY, USA, 2018.

[14] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31, 2018.

[15] J. L. Sabourin, L. R. Shores, B. W. Mott, and J. C. Lester. Understanding and Predicting Student Self-Regulated Learning Strategies in Game-Based Learning Environments. *Int. J. of Artificial Intelligence in Education*, 23(1):94–114, Nov. 2013.

[16] W. Schneider. The Development of Metacognitive Knowledge in Children and Adolescents: Major Trends and Implications for Education. *Mind, Brain, and Education*, 2(3):114–121, 2008.

[17] D. H. Schunk and B. J. Zimmerman. Self-Regulation and Learning. In *Handbook of Psychology, Second Edition*. American Cancer Society, 2012.

[18] J. R. Segedy, J. S. Kinnebrew, and G. Biswas. Using Coherence Analysis to Characterize Self-Regulated Learning Behaviours in Open-Ended Learning Environments. *Journal of Learning Analytics*, 2(1):13–48, May 2015.

[19] D. Stipek, S. Recchia, and S. McClintic. Self-evaluation in young children. *Monographs of the Society for Research in Child Development*, 57(1):1–98, 1992.

[20] M. Taub and R. Azevedo. Using Sequence Mining to Analyze Metacognitive Monitoring and Scientific Inquiry based on Levels of Efficiency and Emotions during Game-Based Learning. *Journal of Educ. Data Mining*, 10(3):1–26, Dec. 2018.

[21] P. H. Winne and R. S. J. d. Baker. The Potentials of Educ. Data Mining for Researching Metacognition, Motivation and Self-Regulated Learning. *Journal of Educ. Data Mining*, 5(1):1–8, May 2013.

[22] B. J. Zimmerman. Investigating Self-Regulation and Motivation: Historical Background, Methodological Developments, and Future Prospects. *American Educational Research Journal*, 45(1):166–183, Mar. 2008.

Automatic Domain Model Creation and Improvement

Philip I. Pavlik Jr., Luke G. Eglington, and Liang Zhang
University of Memphis
ppavlik, lgglngtn, lzhang13@memphis.edu

ABSTRACT

We describe a data mining pipeline to convert data from educational systems into knowledge component (KC) models. In contrast to other approaches, our approach employs and compares multiple model search methodologies (e.g., sparse factor analysis, covariance clustering) within a single pipeline. In this preliminary work, we describe our approach's results on two datasets when using 2 model search methodologies for inferring item or KCs relations (i.e., implied transfer). The first method uses item covariances which are clustered to determine related KCs, and the second method uses sparse factor analysis to derive the relationship matrix for clustering. We evaluate these methods on data from experimentally controlled practice of statistics items as well as data from the Andes physics system. We explain our plans to upgrade our pipeline to include additional methods of finding item relationships and creating domain models. We discuss advantages of improving the domain model that go beyond model fit, including the fact that models with clustered item KCs result in performance predictions transferring between KCs, enabling the learning system to be more adaptive and better able to track student knowledge.

Keywords

Knowledge component model, domain model, learning transfer

1. INTRODUCTION

This paper describes preliminary progress to create a multimethod pipeline to determine the knowledge model (or domain model) that allows the most accurate prediction of performance in an adaptive learning system using a quantitative model of practice. A broad use of quantitative models of practice is to predict performance and make pedagogical decisions [1; 2]. To do this effectively, models typically assign sets of problems or items specific skill tags (often called knowledge components, or KCs). Having such an identification allows a system to monitor which skills have been learned and which need more practice. The matrices representing these item assignments to skills are called Q-matrices [4]. Because the act of tracing student learning is so important for pedagogy, the assignment of items to KCs is crucially important for systems to make pedagogical decisions. Without such an assignment, a system would conceivably need to schedule all items for practice to ensure mastery, so the assignment or "domain model" must be accurate for a system to perform well. Improvements in the domain model may result in better pedagogical decisions in a system. This paper describes a more general approach to improve these critical domain models, a tradition that has included much prior work [3; 5; 14; 15; 19; 20; 22].

In addition to improving domain models, we highlight how these methods may alter how many quantitative models work by enabling models where multiple knowledge components can influence a single practice trial. While such models are not new [13], specifying them with experts is time consuming and error prone. Despite this difficulty, domain models that include the potential for multiple KCs affecting a single performance also typically capture transfer when a shared KC is used in multiple items. In addition to making models more accurate, this transfer has large potential impacts on pedagogy in a complex adaptive instructional system since transfer in an adaptive system means that a KC's performance may bias the selection of other items that share KCs. This transfer will occur because the shared KC will affect the item predictions, making items sharing a KC more or less likely to be practiced.

2. ANALYSIS METHOD

We have developed an automatic domain model improvement algorithm with a highly configurable analysis pipeline.

2.1 Step 1

First is the preprocessing stage. In this stage, some matrix-based method will produce some featural vector of information representing each item. There are two ways this method might process the data from an educational system, either all at once or sequentially in the order the student saw the items. In the first case, this would include methods such as SPARFA-Lite, which assumes one observation for each skill for each student [14]. Our example in this paper uses the SPARFA-Lite model and a simpler model based on covariance clustering [20]. For our examples, Step 1 meant averaging KCs performances for each subject to get a student performance by KC. More advanced methods such as tensor analysis can proceed with sequential data for each student. However, this is future work not presented here.

2.2 Step 2 Infer Feature Matrix

In this step, the method is applied to the data to get some matrix. Currently, the pipeline has two possibilities at this stage, but we plan to include multiple methods in future work as we look to our long-term goal of building a shareable tool for the EDM community.

2.2.1 Covariance Clustering

Developed by Pavlik, Cen, Wu, and Koedinger [20], covariance clustering is a method to describe how each item or existing KC in a domain model is related to all other items or KCs (using a measure of conditional log odds to represent covariance). This method computes a vector for each item representing the conditional

Philip I. Pavlik Jr., Luke Eglington and Liang Zhang "Automatic Domain Model Creation and Improvement". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 672-676. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

probability table for success and failure for the items/KCs relative to all other items/KCs. The pairwise relationships between each vector are similar to the relationships inferred in POKS (Partial Order Knowledge Spaces, [7; 8]), a method related to Falmagne's work [10; 11]. An advantage of covariance clustering is that it characterizes each pairwise relationship between items/KCs in terms of the relationship with all other items/KCs. Pavlik et al. [20] used clustering to establish how to group items by using this KC/item relational vector as the set of features.

2.2.2 SPARFA-Lite

Developed at Rice University by Lan, Studer, and Baranuik [14], SPARFA is a factor analysis method to extract factors from binary-valued data. It provides an association matrix similar to a dAFM Q-matrix with graded associations of concepts with items. The "Lite" version simplifies the method by reducing the parameters and allowing automatic inference of the optimal number of concepts. This method works differently than dAFM, but it provides similar results, allowing for direct comparisons. Also, the ability to infer the optimal number of concepts may be a useful constraint when applying other algorithms.

2.3 Step 3 Cluster Principal Components of Features

In this step, the information matrix is clustered using some method to group items into clusters. Our current implementation first uses RSVD (Randomized Singular Value Decomposition) to simplify the information matrix. We see from the pattern in the results section how the quantity of RSVD components influences the clustering result. We are currently using K-means clustering for clustering, so our search is across both RSVD number of components (N) and K for the number of K means clusters.

For this step, we have also done considerable experimentation with the cmeans fuzzy clustering method, which provides a 0 to 1 index of how strongly each KC is associated with each cluster. Typically, we have used this by specifying a threshold (which can be optimized with search) over which an item belongs to each cluster or not. This assignment allows for membership in multiple clusters, which means that unlike the method in Step 2.4, the item is assigned to potentially many clusters. Typically, when we use this method, we have weighted the effect of prior practice for the KC clusters according to the number of KC clusters involved in a performance. This weighting is not necessary for the simpler K-means implementation since the added KC column only assigns each KC to 1 KC cluster.

2.4 Step 4 Fit with New Model

We used the new model as an overlay such that we created a column with the cluster id for each KC for each trial. This overlay procedure means that while the KC and clusters are independent, practicing an item may affect other items if they share a cluster. To do this, we first describe our starting model, which was simply PFA (Performance Factors Analysis, [17]) using the logarithms of practice counts for successes and failure (adding 1 to each to permit the logarithms). Where θ values are Student ability and KC difficulty respectively, and *S* and *F* represent the count of prior success and failures for the KC *j* for the student *i*.

$$logit(p_{ijt}) = \beta_1 \log_e S_{ij} + \beta_2 \log_e F_{ij} + \theta_i + \theta_j$$

The new model was defined using cluster-id (c) as a KC in an additive compensatory model. Prior research suggests such compensation among KCs works well for prediction [6; 16].

$$logit(p_{ijt}) = \beta_{1j} \log_e S_{ij} + \beta_{2j} \log_e F_{ij} + \beta_{3c} \log_e S_{ic} + \beta_{4c} \log_e F_{ic} + \theta_i + \theta_j$$

Two versions of this model with clusters were compared; the first version was as described, and the second version was a control condition where the cluster column was sampled at random from the Q-matrix. This control condition should exhibit the same amount of overfitting due to adding parameters but none of the benefit of a coherent clustering solution. These models are compared using 2 to N components and 2 to K clusters by iterating to Step 3 to search a space of models.

In the context of our future work, we plan to allow users of our tool to specify candidate models with different configurations and terms using a logistic knowledge tracing R package freely available [18]. It is possible that different learner models may be implemented at this step since the Q-matrices we are creating may be used in many types of learner models.

2.5 Step 5 Splitting and Merging

Just as steps 3 and 4 may iterate to find optimal K and N, steps 4 and 5 may iterate to refine the model in Step 4. This step describes our future planning for a tool to optimize Q-matrix type models of knowledge domains.

Splitting takes the original KC model and uses the KC model from the clustered features to determine hypotheses for how KCs might be split. So if a KC in the original model is in 2 clusters, the model would test whether that that was best represented by the default model (include the effect of the cluster and the KC for each KC) or whether the cluster was unnecessary and the fit was just as good by splitting the KC into two different KCs and dropping the effect of the cluster KC. Further, we could also test whether the specific clusters proposed for each KCs even improves fit by removing them entirely as a third hypothesis. Two of these three possibilities correspond to Learning Factors Analysis (LFA) [5], and the third (including the cluster instead of using a split) advances the approach.

Merging uses the cluster model like LFA, but instead of splitting KCs, the clusters are used to evaluate three hypotheses about whether existing KCs can be merged into a single KC. One hypothesis is that the KCs are best represented as separate but influence each other through the shared cluster membership. The second hypothesis is that the specific cluster was unnecessary, and the two KCs should be merged into 1 KC. Finally, the third hypothesis is that the 2 KCs are separate and that the cluster predictor should be omitted.

Step 5 is similar to backward and forward stepwise regression methods, and so it is clear this method would be very likely to cause overfitting due to the way it will tailor the model term to capture the data iteratively. To prevent this problem, the solutions produced are robustly cross-validated. By tuning the model to maximize cross-validation accuracy, we aim to find quantitative thresholds for when to add or subtract terms from the model with a result that is efficient and parsimonious.

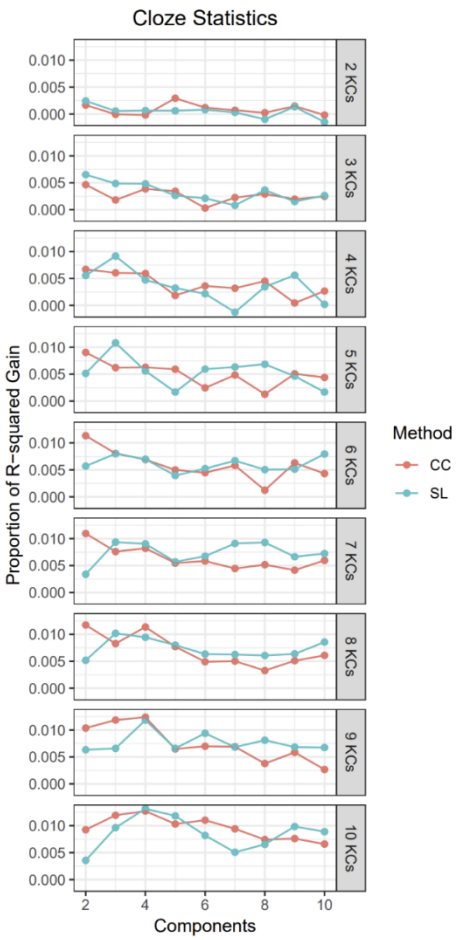


Figure 1. Changes in fit including differing numbers of additional clusters for Cloze dataset using covariance clustering (CC) or SPARFA-lite (SL).

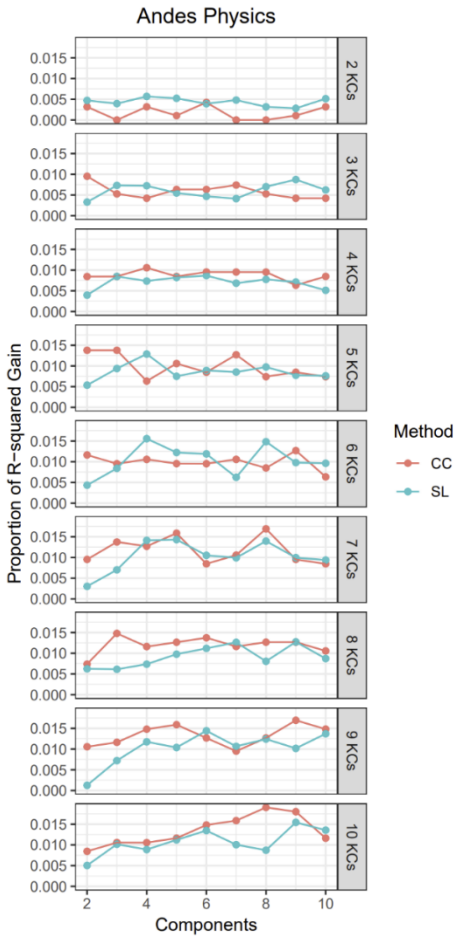


Figure 2. Changes in fit including differing numbers of additional clusters for Andes Physics data using covariance clustering (CC) or SPARFA-lite (SL).

3. DATASETS

The statistics cloze dataset included 58,316 observations from 478 participants who learned statistical concepts by reading sentences and filling in missing words. Participants were adults recruited from Amazon Mechanical Turk. There were 144 KCs in the dataset, derived from 36 sentences, each with 1 of 4 different possible words missing (cloze items). The number of times specific cloze items were presented was manipulated, and the temporal spacing between presentations (narrow, medium, or wide). The post-practice test (filling in missing words) could be after 2 minutes, 1 day, or 3 days (manipulated between students). The stimuli type, manipulation of spacing, repetition of KCs and items, and multiple-day delays made this dataset appropriate for evaluating model fit to well-known patterns in human learning data (e.g., substantial forgetting across delays, benefits of spacing). The dataset was downloaded from the Memphis Datashop repository.

In the Andes dataset, 66 students learned physics using the Andes tutoring system, generating 345,536 observations. Participants were given feedback on their responses as well as solution hints. Additionally, participants were asked qualitative "reflective" questions after feedback (for more details, see [12]).

4. RESULTS

Figures 1 and 2 show the result for the two datasets. The proportion of R-squared gain indicates the improvement in R-squared for the true clustered model compared to the random comparison R-squared model as a proportion of the random comparison R-squared model. Because of the result's preliminary nature, we have not been able to produce smoothed figures through cross-validation. However, the results consistently show beneficial effects. In general, both methods have similar accuracy.

Both methods can achieve similar improvements via different parameters. However, it does appear that the efficacy of the methods differs somewhat across datasets. Covariance clustering found the best solution in the Andes dataset, with SPARFA-Lite having the best solution in the Cloze dataset. This preliminary result suggests that applying multiple approaches to the same dataset may be advisable, especially when the underlying domain structure is unknown. Different domain modeling algorithms may differ in their ability to detect this underlying domain structure.

To understand better the results shown in Figures 1 and 2 we can query the model for the parameters for the cluster KCs to confirm that they are meaningful due to the structure of PFA. Normally we would expect the cluster KC coefficients for success and failure to be more different if the model was labeling real KCs since it is typically the case that successes predict future success more than failures. Indeed, test comparison shows exactly this pattern; for

example, considering the model with 10 components and 10 KCs in for the Physics data with covariance clustering applied, we see the success is .56 higher than failure. In contrast, for the randomized model, the value was .12 higher for success than failure (best explained as the overfitting we might expect for such a mechanism).

5. DISCUSSION

In the present paper, we described our ongoing work to automate both the process of searching for domain models and the search method (e.g., covariance clustering vs. SPARFA-Lite). Many approaches have been proposed to infer domain [3; 5; 14; 15; 19; 20; 22], but there has been little comparison. However, comparison among approaches is important because their different underlying assumptions and limitations will interact with the learning domain's true underlying structure. For example, if the learning domain is calculus, various prerequisite skills from other branches of mathematics may be necessary (e.g., algebra, trigonometry). In other domains, learning one KC before another may enhance learning but not be required. Learning how to compute a sample's mean may facilitate learning to compute the *median* due to contrasting their different procedures. However, neither is a prerequisite to learn the other. Domains vary in the extent that learning one KC may transfer to another, and the researcher may not have strong theories a priori that could help constrain the KC model search. Thus, choosing one method with specific assumptions and limitations across different knowledge domains may be inadvisable and result in suboptimal KC model solutions.

5.1 Future Plans

5.1.1 Additional domain model methods

There are several methods we hope to include in the system to analyze student data to produce the inference matrix, for example:

dAFM - Developed at Berkeley by Pardos and Dadu [15] and shown to improve the Piech [21] deep knowledge tracing algorithm. This method is a deep learning model that uses backpropagation to infer a Q-matrix type representation with graded skill assignments instead of binary assignments. The authors show how the model is a continuous neural network generalization of the AFM model used in the LFA method [5].

Tensor factorization – We have also been working with implementations of tensor factorization. Tensors allow the solution to integrate multiple sources of data, including a representation of time in the sequence of practice.

These methods may be more accurate because they allow the representation of sequence to capture order effects in the model that may be due to learning. However, another view might be that it makes it more vulnerable to the selection effects that led to particular practice sequences. In other words, domain model search algorithms that are sensitive to effects over time may be more likely to incorporate artifacts due to pedagogical decision rules (e.g., "drop item from practice after N successes"). For instance, in systems in which items are dropped from practice after a few successes (e.g., Assistments), the sequential order and temporal spacing will be different than in practice schemes in which items are not dropped from practice (e.g., [9]). In short, domain model extraction from datasets that were generated by an adaptive learning system will be influenced by the decision rules inherent to that system.

5.1.2 An ensemble approach to address individual model search limitations

We also intend to allow multiple approaches to be allowed within a single KC model development pipeline. For instance, approaches like dAFM have shown promise to improve KC models but require an initial KC model. However, this apparent limitation is only a problem if the goal is to find a single approach that resolves the problem of KC modeling. Instead, the goal can be reoriented towards finding the best ensemble and ordering of approaches that can be used in order to develop an optimal KC model. As an example, an optimal KC model may be created by making an initial model with SPARFA-Lite, followed by a final model using dAFM. Requiring a starter model is only a limitation if complementary approaches cannot be combined.

5.1.3 Integrating with learner model development

Learner models and domain models are strongly interdependent but frequently developed and refined independently. This separation probably limits progress on both fronts. Using relatively simple learner models when searching for improved domain models may lead to misleading results if the chosen learner model does not accurately represent learning, forgetting, transfer, and other important learning factors. Similarly, developing learner models without considering the chosen KC model's plausibility may lead to spurious results. Recently, we developed a framework to facilitate learner model development named Logistic Knowledge Tracing [18]. We aim to integrate automated KC model search and refinement into the LKT framework.

5.1.4 Representing transfer does more than improve model fit

Representing transfer among KCs can have significant pedagogical consequences that will not be apparent from model fit metrics (e.g., reduced RMSE, increased AUC). For instance, imagine a student is learning three items (A, B, and C). If the domain model considers A and B to be related because they share KC, practicing item A will influence both *when* and *how much* B is practiced. Depending on the strength of the transfer, practicing A may result in B being practiced being, being practiced before C, being practiced after C, or not being practiced until much later when forgetting has occurred (if the learner model assumes forgetting). The entire order of practice may change.

Another issue is the efficiency effect of such transfer. Consider that if the three items are independent, students may practice all three as necessary for mastery. In contrast, if item A affects item B through a shared KC, it will increase or reduce the amount of practice needed for mastery of B, which can reduce costly overpractice. In short, accounting for transfer among KCs may greatly improve practice *efficiency*, which may not be apparent when comparing domain models in terms of model fit metrics (e.g., RMSE, AUC, AIC). Ultimately, comprehensive evaluation of new KC models requires simulations or experiments to determine their effects on how practice is scheduled within an adaptive learning system. This need comes from how a new KC model may interact with pedagogical decision rules (e.g., mastery learning) and learner models (e.g., BKT, PFA) within an adaptive learning system to change the *sequence* of practice (e.g., due to quantifying transfer among items differently). These changes to the sequence may have significant impacts on student learning.

6. ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation Learner Data Institute (NSF #1934745) projects and a grant from the Institute of Education Sciences (ED #R305A190448).

7. REFERENCES

[1] Atkinson, R.C., 1972. Ingredients for a theory of instruction. *American Psychologist 27*, 10 (Oct), 921-931. DOI= http://dx.doi.org/http://doi:10.1037/h0033572.

[2] Atkinson, R.C., 1972. Optimizing the learning of a second-language vocabulary. *Journal of Experimental Psychology 96*, 1, 124-129.

[3] Barnes, T., 2005. The Q-matrix method: Mining student response data for knowledge. In *American Association for Artificial Intelligence 2005 Educational Data Mining Workshop*, J. Beck Ed. AAAI Press, Pittsburgh, PA, USA, 39-46. DOI= http://dx.doi.org/http://doi:10.1.1.531.3631.

[4] Birenbaum, M., Kelly, Anthony E., and Tatsuoka, Kikumi K., 1992. *Diagnosing knowledge states in algebra using the rule space model*. Educational Testing Service.

[5] Cen, H., Koedinger, K.R., and Junker, B., 2006. Learning Factors Analysis - A general method for cognitive model evaluation and improvement. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* Springer Berlin / Heidelberg, 164-175.

[6] Cen, H., Koedinger, K.R., and Junker, B., 2008. Comparing two IRT models for conjunctive skills. In *Proceedings of the Proceedings of the 9th International Conference on Intelligent Tutoring Systems* (Montreal, Canada2008), 796-798.

[7] Desmarais, M.C., Meshkinfam, P., and Gagnon, M., 2006. Learned student models with item to item knowledge structures. *User Modeling and User-Adapted Interaction 16*, 5, 403-434.

[8] Desmarais, M.C., Pu, X., and Blais, J.-G., 2007. Partial Order Knowledge Structures for CAT Applications. In *Proceedings of the 2007 GMAC Conference on Computerized Adaptive Testing* (2007).

[9] Eglington, L.G. and Pavlik Jr, P.I., 2020. Optimizing practice scheduling requires quantitative tracking of individual item performance. *npj Science of Learning 5*, 1 (Oct. 15,), 15. DOI= http://dx.doi.org/10.1038/s41539-020-00074-4.

[10] Falmagne, J.-C., Doignon, J.-P., Cosyn, E., and Thiery, N., 2003. The assessment of knowledge in theory and in practice. *Institute for Mathematical Behavioral Sciences Paper 26*.

[11] Falmagne, J.-C., Koppen, M., Villano, M., Doignon, J.-P., and Johannesen, L., 1990. Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review 97*, 2 (Apr), 201-224.

[12] Katz, S., Connelly, J., and Wilson, C., 2007. Out of the lab and into the classroom: An evaluation of reflective dialogue in Andes. *Frontiers in Artificial Intelligence and Applications 158*(Jun.), 425-432. DOI= http://dx.doi.org/http://doi:10.5555/1563601.1563669.

[13] Koedinger, K.R., Pavlik Jr., P.I., Stamper, J., Nixon, T., and Ritter, S., 2011. Avoiding Problem Selection Thrashing with Conjunctive Knowledge Tracing. In *Proceedings of the 4th International Conference on Educational Data Mining*, M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero and J. Stamper Eds., Eindhoven, the Netherlands, 91–100.

[14] Lan, A.S., Studer, C., and Baraniuk, R.G., 2014. Quantized Matrix Completion for Personalized Learning. In *Proceedings of the 6th International Conference of Educational Datamining*.

[15] Pardos, Z. and Dadu, A., 2018. dAFM: Fusing psychometric and connectionist modeling for Q-matrix refinement. *Journal of Educational Data Mining 10*, 2 (Oct.), 1-27. DOI= http://dx.doi.org/http://doi:10.5281/zenodo.3554689.

[16] Pardos, Z.A., Beck, J.E., Ruiz, C., and Heffernan, N.T., 2008. The composition effect: Conjunctive or compensatory? An analysis of multi-skill math questions in ITS. In *1st International Conference on Educational Data Mining*, R.S. Baker, T. Barnes and J.E. Beck Eds., Montreal, Canada, 147-156.

[17] Pavlik Jr, P.I., Cen, H., and Koedinger, K.R., 2009. Performance factors analysis: A new alternative to knowledge tracing. In *14th International Conference on Artificial Intelligence in Education*, V. Dimitrova, R. Mizoguchi, B.D. Boulay and A. Graesser Eds., Brighton, England.

[18] Pavlik Jr, P.I., Eglington, L.G., and Harrell-Williams, L.M., 2021, preprint. Logistic Knowledge Tracing: A constrained framework for learner modeling. *arXiv.org*.

[19] Pavlik Jr., P.I., Cen, H., and Koedinger, K.R., 2009. Learning factors transfer analysis: Using learning curve analysis to automatically generate domain models. In *Proceedings of the 2nd International Conference on Educational Data Mining*, T. Barnes, M.C. Desmarais, C. Romero and S. Ventura Eds., Cordoba, Spain, 121–130.

[20] Pavlik Jr., P.I., Cen, H., Wu, L., and Koedinger, K.R., 2008. Using item-type performance covariance to improve the skill model of an existing tutor. In *Proceedings of the 1st International Conference on Educational Data Mining*, R.S. Baker and J.E. Beck Eds., Montreal, Canada, 77–86.

[21] Piech, C., Spencer, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., and Sohl-Dickstein, J., 2015. Deep Knowledge Tracing. *arXiv preprint arXiv:1506.05908*.

[22] Sahebi, S., Lin, Y.-R., and Brusilovsky, P., 2016. Tensor Factorization for Student Modeling and Performance Prediction in Unstructured Domain. *International Educational Data Mining Society*.

Automated Classification of Visual, Interactive Programs Using Execution Traces

Wengran Wang
North Carolina State University
wwang33@ncsu.edu

Gordon Fraser
University of Passau
gordon.fraser@uni-passau.de

Tiffany Barnes
North Carolina State University
tmbarnes@ncsu.edu

Chris Martens
North Carolina State University
crmarten@ncsu.edu

Thomas Price
North Carolina State University
twprice@ncsu.edu

ABSTRACT

Offering students immediate, formative feedback when they are programming can increase students' learning outcomes and self-efficacy. However, visual and interactive programs include dynamic user input and visual outputs that change over time, making it difficult to automatically assess students' code with traditional functional tests to offer this feedback. In this work, we introduce Execution Trace Based Feature Engineering (ETF), a feature engineering approach that extracts sequential patterns from execution traces, which capture the runtime behavior of students' code. We evaluated ETF on 162 students' code snapshots from a Pong game assignment in an introductory programming course, on a challenging task to predict students' success on fine-grained rubrics. We found that ETF achieves an average F_1 score of 0.93 over 10 grading rubrics, which is 0.1–0.2 higher than a high-performing syntax-based code classification approach from prior work. These results show that ETF has strong potential to be used for code classification, to enable formative feedback for students' visual, interactive programs.

Keywords

execution traces, feature engineering, computer science education, code classification, formative assessment

1. INTRODUCTION

Real-time, formative feedback promotes students' learning gains and self-efficacy [7, 3, 12, 18]. To provide such formative feedback in real-time, CS instructors commonly write test cases, allowing students to run their code against these test cases when programming [9, 5, 6, 4]. However, visual, interactive programming projects, such as creating apps and games [16], include dynamic user interactions, and visual outputs that change over time, making it challenging to use test cases to assess these programs [14, 13, 20].

In contrast to test case-based approaches, data-driven methods allow instructors to offer formative feedback by grading a smaller set of programs instead of writing test cases [11, 21, 10, 22]. These methods start with transforming code into input vectors using **feature engineering**, typically by extracting syntax elements from an abstract syntax tree (AST), where nodes and their children correspond to specific code elements (e.g., if statements). However, when applying these syntax-based feature extraction techniques to classify programs, prior work showed mixed results, which are often not high enough to ensure the quality of student feedback [11, 1, 2]. Some prior work has used **execution traces** to classify students' sorting programs based on their specific strategies, and has shown that execution-trace-based classification achieved higher accuracies than a syntax-based classification approach [8]. However, no prior work has conducted feature extraction on the execution trace of visual, interactive programs, which include dynamic user interactions and various changing outputs. In this work, we extract features from execution traces that capture the runtime behavior of visual, interactive programs. We designed an execution trace-based feature engineering approach (ETF) to transform students' source code into feature vectors, for classification algorithms to build models based on rubric-based labels (e.g., the presence of a key-triggered movement). We evaluated ETF by classifying 162 students in-progress and submitted code snapshots. We found it to achieve high prediction performance with an average of 0.93 F_1 score over 10 grading rubric items, which is 0.1–0.2 higher than a high-performing syntax-based code classification approach. Our work has the following contributions: 1) We designed and implemented a novel, execution trace-based feature engineering (ETF) approach to extract temporal patterns in students' visual, interactive programs; 2) We evaluated the ETF approach on students' code snapshots for a widely-used, representative visual, interactive program assignment.

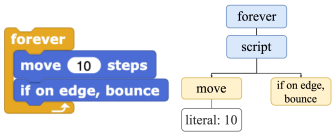


Figure 1: A horizontal n(2)-Gram (in yellow) and a vertical n(2)-Gram (in blue).

2. RELATED WORK

Syntax-based approaches extract patterns inside a code AST, use the presence or absence of a feature, or the count of the feature to generate input vectors. As an example,

Wengran Wang, Gordon Fraser, Tiffany Barnes, Chris Martens and Thomas Price "Automated Classification of Visual, Interactive Programs Using Execution Traces". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 677-681. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

a) Automated Program Execution

clock time	step	ball		
		position	direction	TouchSprite
1.974s	119	x:194, y:14	146	{ }
1.992s	120	x:200, y:6	146	{ }
2.019s	121	x:205, y:-3	214	{paddle}
2.024s	122	x:183, y:-35	214	{ }

b) Dumping Execution Traces

Figure 2: Step 1: Generating Execution Traces.

we explain a recently-applied AST n-Gram feature extraction approach [17, 2] by making an analogy to Natural Language Processing (NLP): In NLP, an n-Gram with $n = 1$ is a *1*-Gram feature taken from each word; and an n-Gram feature takes a continuous sequence of n words to extract relationship between words (e.g., orders); in an AST, *1*-Gram features represent each *node* in students' code. And to extract structural relationships, Akram et al. [2] designed the use of n-Grams to represent n-length sequences of code, where a ***vertical n-Gram*** is created by a depth-first search of leaves; a ***horizontal n-Gram*** is created by a breadth-first search of all direct children of each AST nodes (e.g., in Figure 1) [2]. *1*-Gram and n-Gram-based AST feature extraction approaches have both been applied for student code classification tasks. Compared to *1*-Gram feature extractions, prior work has shown that n-Grams provide more predictive features for code analysis. For example, Akram et al. used n-Grams with n ranging from 1 to 4 to extract features, and used a Gaussian Process model to infer scores on 642 students' code pieces, in a block-based programming environment. This achieved an R-square of 0.94, higher than the 0.88 achieved by the baseline *1*-Gram approach [2, 1].

On the other hand, recent work by Paaßen et al. used **execution-trace-based** distance measures to classify programs into different strategies (e.g., bubble sort v.s. Insertion sort), and found that execution- race-based classification achieving 90% accuracy, higher than the 80% accuracy achieved by syntax-based approaches [8]. This shows the potential of using execution traces to classify students' programming code.

3. THE ETF APPROACH

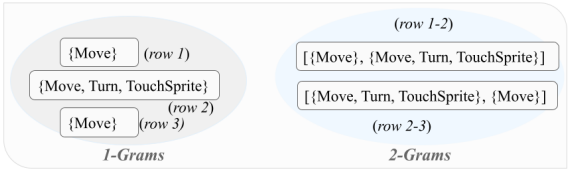
ETF starts from collecting a set of students' programming code, along with a class label for each piece of code, (i.e., positive or negative). ETF is designed for programs that have the following properties: 1) respond to **dynamic user inputs** (e.g., mouse, keyboard). 2) has **object-specific program states**, corresponding to visual output on the screen. 3) Program behaviors can be **a function of time**; and can also **change over time**.

An Example Assignment. As an example, consider the Pong assignment, which consists of a paddle sprite and a ball sprite. The ball moves around the stage [19], and a player can use the keyboard to control the up and down movement of the paddle to catch the running ball. If the paddle catches the ball, the player score increases; but if the paddle misses the ball and the ball hits the back wall, the game ends. ETF conducts feature engineering on such visual, interactive programs in four steps, described below.

step window	ball				
	Move?		Turn?	TouchSprite?	
119 - 120	✓	{right, down}	✗	✗	
120 - 121	✓	{right, down}	✓	✓	{paddle}
121 - 122	✓	{left, down}	✗	✗	

Summary Trace:
row 1: {Move({right, down})}
row 2: {Move({right, down}), Turn, TouchSprite({paddle})}
row 3: {Move({left, down})}

a) Step 2: Generating Summary Trace



b) From Step 3: 1-Grams and 2-Grams

Figure 3: Step 2&3: summarizing traces & generating features.

3.1 Step 1: Generating Execution Traces

Visual, interactive programs include program states that can be represented as ***properties*** that change over time. For example, in Snap*!*, these properties can include: 1) ***Time***: how much milliseconds has passed from the start of program execution; 2) *inputs*: including `KeysDown` (which key is pressing); `MouseDown` (if mouse pressing); `MouseX` and `MouseY` (x, y positions of mouse) 3) *global variables*: the names (`Var.Name` and values `Var.Value`) of global variables; 4) *sprite-specific properties*: properties that are related to specific sprites, such as (`x`, `y`) (x, y coordinates); `dir` (directions); `TouchSprite` (which sprites the current sprite is touching); `TouchEdge` (which stage edge the sprite is touching); `size` (sprite size); `OffStage` (whether the sprite is moved out of the stage); the names (`Var.Name`) and values `Var.Value` local variables. The dumped execution trace tables use sprite names to label sprite-specific properties (e.g., to distinguish `ball.x` from `paddle.x`[1]).

Systems such as Snap*!* and Scratch make use of ***step functions*** to update the above properties based on the current properties and the current user inputs [14, 19]. We instrumented the step function in Snap*!* with a trace logging tool, so that with each Step, it adds a row in an ***execution trace table*** with the properties listed above, and dumps the trace table at the end of the execution. Figure 2 gives an example of a part of the *execution trace table*, in which one row logs one discrete Step created by the step function, with each entry maps to a *property* (i.e., a concrete program state).

3.2 Step 2: Summarizing Traces

ETF algorithm scans through the execution trace table in a sliding window of multiple steps (default as 2), apply a ***Trace Abstraction Function (TAF)***. The TAF looks for properties based on ***candidate properties***, and only returns properties that were found in the sliding window as a *summarized property set*. A *candidate property* can be an ***abstract property***, describing the changes between steps in the execution trace, such as movement and variable change; A *candidate property* can also be an original trace *property* which were already recorded in the execution trace. In each sliding window, the TAF function returns a *summarized property set* that includes all found properties, for example, in the step window 120-121 of Figure 3, all candidate properties were found because there is a change in po-

[1]ETF uses these properties to summarize trace and generate features (Section 3.3). To allow comparison across students, sprites need to have consistent labels across student programs.

sition (`Move`), a change in direction (`Turn`), and a non-empty `TouchSprite` column in Step 121. This creates a *summarized property set* {`Move`, `Turn`, `TouchSprite`}, shown in Row 2 of the *summary trace* (Figure 3). In addition, TAF's candidate properties also include possible types of ***parameters***, which describe detailed information of the property.

For Pong, ETF used 9 types of ***candidate properties***. Except 2 program state properties: (`KeysDown` and `ChangeVar`), the rest 7 are sprite-specific properties and are labeled with the sprite names (e.g., `Paddle.Move`). Among the 9 *candidate properties*, 4 were original trace *properties*, directly returned when the corresponding property in the last step of the sliding window is non-empty: `KeysDown`, `TouchSprite`, `TouchEdge`, and `OffStage`, using the same parameters with the corresponding execution trace entry at the last step of the sliding window, explained in Section 3.1. Others are 5 *abstract properties*, that only checks if a property changes between the first and last step of the sliding window (and if has middle, omit those middle ones). 1) `Move`$\langle\leftarrow, \rightarrow, \uparrow, \downarrow\rangle$. Returned when a sprite position changes. Its parameters are the direction toward which the sprite moves. 2) `Turn`. Returned when a sprite changes direction. 3) `ChangeSize`$\langle$`+, -`$\rangle$. Returned when the Sprite changes its size to bigger (+) or smaller (-). 4) `ChangeVar`$\langle$`variable names`$\langle$`+, -`$\rangle\rangle$. Returned when a global variable's value has been changed to bigger (+) or smaller (-). 5) `ChangeLocalVar`$\langle$`variable names` $\langle$`+, -`$\rangle\rangle$. Returned when a local variable's value has been changed to bigger (+) or smaller (-).

3.3 Step 3: Generating n-Gram-based Features

ETF next transforms *summary trace* created by Step 2 into a set of features using n-Gram-based approach, where an n-Gram takes a contiguous sequence of n items in data (Figure 3). ETF extracts features of 4 types: 1) ***1*-Grams**, extracting **simultaneous** behaviors, taken from each row of the summary trace. 2) ***2*-Grams**, connecting adjacent *1*-Grams sequentially; 3) **Power Sets.** We extract n-Grams of not only the full property set in each row of the *summary trace*, but also of subsets of the property set, such as the 2-set of just `Move` and `Turn` from t *1*-Gram {`Move`, `Turn`}. When constructing power sets for *2*-Grams, we apply the power set on the types of properties that are possible in this *2*-Gram. 4) For all the n-Grams extracted above, we keep both ***non-parameterized n-Grams***, where we do not record the parameters, as well as ***parameterized n-Grams***, where each property would include its parameters when they were logged in the summary trace. Next, ETF collects distinct features from all students' feature sets as the ***full feature set***, which consists of distinct features from all students.

3.4 Step 4: Filtering Features

Merging duplicate features and removing rare features. Based on the *full feature set* generated from Step 3, if features have the exact same distribution among programs, the ETF algorithm then merges these features as one feature; and it calculates the support of each feature based on the proportion of student programs that include this feature, and remove features that have support lower than a certain threshold, determined by a hyperparameter tuning process, described in Section 4.2.

Generating $\vec{x}$ vectors. After generating, merging duplicates, and removing rare features, we use the resulting feature set as the independent variables, and for each student program, we use 1 as representing the presence of a feature in the student program (i.e. the n-Gram appeared at least once in their abstracted execution trace), 0 as the absence of the feature, and generate 0-1 digitized $\vec{x}$ vector for each student's code snapshot, which is used as vector input for a classification model.

4. EVALUATION

We investigate our research question: **How accurately does ETF perform rubric-based code classification of students in-progress and submitted code, and how does this compare to syntax-based approaches?**. We first a) compare performance of ETF features and syntax-based features **across models**; and next b) compare ETF features and syntax-based features **across rubrics** on a fixed model. Our analysis of a) and b) follows the same procedure, where we started by generating ETF and syntax-based features separately (Section 4.1). We next performed the same feature filtering, training and evaluation procedure on the features we created (Section 4.2).

Dataset. We evaluated ETF on 42 students' 162 code snapshots for a Pong game assignment, sampling student code snapshots at 10 minutes (42), 20 minutes (40) and 30 minutes (38) of work, as well as their final submissions (42), on 10 target evaluation items: key_up/down, upper/lower_bound, space_start, edge_bounce, paddle_bounce, paddle_score, reset_score, reset_ball. A detailed description and the prevalence of the data can be found in our prior work [19].

4.1 Generating Features

Generating ETF features. We used the procedure described in the Step 1 – 3 (Section 3.1–3.3) of the ETF approach to generate ETF features. We first automatically run the program based on inputs. To ensure coverage, we used the same inputs (up/down arrow key, follow/evade ball) as our in prior work [19], defined using SnapCheck. For each program snapshot, we re-executed the program 5 times. Each run of student programs generated one execution table.

Generating AST *n*-Gram and *1*-Gram Features. We first compare ETF it with a representative, syntax-based feature extraction approach that has performed well in prior evaluations by using the AST n-Gram feature extraction approach [2]. Similar to Akram et al.'s work, we extracted all n-Grams from all ASTs, using $n = 1$ to 5 for *vertical n-Grams*, and $n = 1$ to 4 for *horizontal n-Grams* (explained in Section 2). Similar to many AST feature extraction approaches [10, 11, 22], we used a single label for all literals (`literal`).

4.2 Feature Filtering & Evaluation

We applied the same feature filtering and evaluation to the ETF, AST n-Gram, and AST *1*-Gram features:

Feature Filtering. For fairness of comparison, after collecting features, we used the Step 4 from the ETF algorithm to filter features for all ETF, AST n-Gram, and AST *1*-Gram features. For each type of the three features, we

Table 1: F1 scores of AST 1-Grams, AST n-Grams, and ETF Features, over different models.

	AST 1-Grams	AST n-Grams	ETF Features
Logistic Regression	0.771	0.779	0.932
AdaBoost	0.78	0.78	0.922
Random Forest	0.763	0.773	0.926
MLP	0.764	0.771	0.908
Gaussian Process	0.739	0.728	0.923
SVM	0.759	0.771	0.93

started by using ETF to automatically merge duplicate features (Step 4.a), and remove features that have support smaller than a certain threshold in the training set (Step 4.b). The threshold is set as a hyperparameter (tuned as described below).

Classification Models. To ensure that our comparison was not model-specific, we used 6 different models on the feature set: Logistic Regression, AdaBoost, Random Forest, Multi-layer perceptron (MLP), Gaussian Process, and SVM. Among them, the Gaussian Process model with an RBF kernel was also employed by Akram et al., and has shown to be the best performing model in the rubric-based performance inference task that they have applied [2].

Training & Evaluation. We employed 10-fold cross-validation to evaluate how accurately these different features predict the rubric-based performance. Within each round of cross-validation, we used another 2-fold cross-validation to tune the hyperparameters (i.e. nested cross-validation [15]). For all models, we included a minimum feature support threshold hyperparameter, T, below which we exclude the feature (e.g. ETF feature or AST n-Gram feature) from the final feature set, with the minimum support threshold as a hyperparameter, tuned based on 5 values: $\{0, 5\%, 10\%, 15\%, 20\%\}$. Additionally, since different feature extraction approaches may perform best with different values of model-specific hyperparameters, we also tuned hyperparameters for each classification models, based on the following values: *Logistic regression*: with penalty in {L1, L2}; *Random Forest*: with n_estimaters (i.e., number of trees in the forest) in $\{100, 200, 300, 400, 500\}$; *AdaBoost*: with learning rate in $\{0.01, 0.1, 1\}$; *MLP*: with learning rate in $\{0.001, 0.01, 0.1\}$; ' *SVM*: We used a linear kernel, with the regularization parameter (C) in $\{0.01, 0.1, 1, 10, 100\}$; Gaussian Process models optimize kernel hyperparameters during model fitting, we therefore did not tune hyperparameters for the Gaussian Process classifier. The values of the minimum feature support threshold and the model-specific hyperparameters were determined by their F_1 scores in the nested 2-fold cross-validation, based on a grid search on 5*#(model-specific hyperparameter values) possible types of hyperparameter combinations, during each round of the 10-fold cross-validation. Since many of our target labels are imbalanced, the accuracy score offers less information on how well our model performs in predicting target labels. We therefore use F_1 scores to tune hyperparameters. To ensure that data from a given student was not contained in *both* training and testing sets, all cross-validation splits were done on the 42 **students** (instead of on the 162 snapshots).

4.2.1 Results

Comparison Across Models. Using each of the 6 models described above, we predict students' rubric-based performance, calculated its F_1 score among the 162 students' data using 10-fold cross validation, creating one F_1 score for each rubric. We next averaged F_1 scores for each classifier across all rubrics, shown in Table 1. We saw that the AST n-Gram approach performed similar to the *1*-Gram approach (5 in 6 cases), and that ETF features generated F_1 scores that were consistently 0.14 to 0.2 higher than the AST n-Gram features, showing that all classifiers benefited from the execution-trace-based information extracted by the ETF features, with overall F_1 scores between 0.9 and 0.93. This result shows potential for us to make use of ETF features to correctly analyze students' current progress, which should enable automated, formative feedback in future work, to help a student who is stuck in the middle of programming.

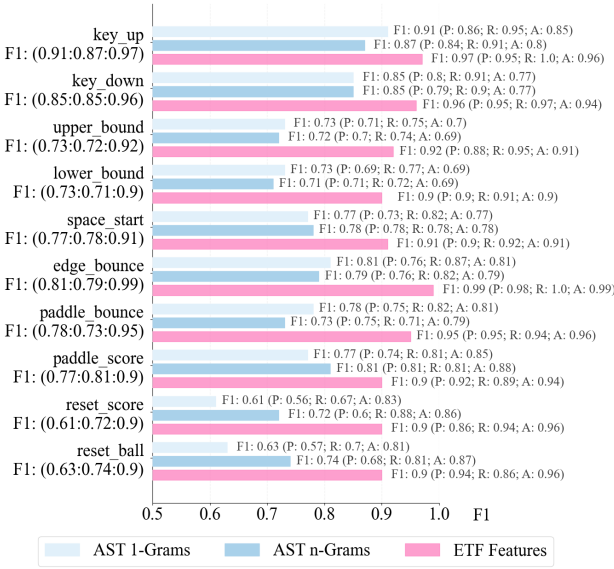


Figure 4: The F_1 score (F1), Precision (P), Recall (R) and Accuracy (A) of ETF, *n*-Gram, and *1*-Gram features on each rubric, using an SVM model. x-axis starts from $F_1 = 0.5$.

Performance Across Rubrics. We next investigate the performance of the three feature extraction approaches **across rubrics**. Since all model show similar trends, we select SVM and present performance on all rubrics in Figure 4. We found 1) The naive AST *1*-Gram features had relatively lower F_1 scores on rubrics that had less prevalence in data (e.g., *reset_score*, *reset_ball*); 2) Comparing to AST *1*-Gram, the AST n-Gram features produced higher F_1 scores for *paddle_score*, *reset_score*, *reset_ball*, showing that AST n-Gram extracted more useful feature for these three rubric items. However, the ETF features performed relatively well across all rubrics, with its F_1 scores ranging from 0.9 to 0.99, showing that ETF features have strong potential to enable formative feedback on a variety of fine-grained, specific rubrics.

In conclusion, we presented a novel, effective approach that extracted useful features from execution traces (ETF), leading to high predictive accuracy. Our results show strong potential for using ETF to monitor student progress and offer automated, formative feedback.

5. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1917885.

References

[1] B. Akram et al. Assessment of students' computer science focal knowledge, skills, and abilities in game-based learning environments. 2019.

[2] B. Akram, W. Min, E. Wiebe, A. Navied, B. Mott, K. E. Boyer, J. Lester, et al. Automated assessment of computer science competencies from student programs with gaussian process regression. In *Proceedings of the 2020 Educational Data Mining Conference*, 2020.

[3] A. T. Corbett and J. R. Anderson. Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 245–252, 2001.

[4] S. H. Edwards and K. P. Murali. Codeworkout: short programming exercises with built-in data collection. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 188–193, 2017.

[5] D. Hovemeyer and J. Spacco. Cloudcoder: a web-based programming exercise system. *Journal of Computing Sciences in Colleges*, 28(3):30–30, 2013.

[6] A. N. Kumar. Explanation of step-by-step execution as feedback for problems on program analysis, and its generation in model-based problem-solving tutors. *Technology, Instruction, Cognition and Learning (TICL) Journal*, 4(1), 2006.

[7] S. Marwan, G. Gao, S. Fisk, T. W. Price, and T. Barnes. Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, pages 194–203, 2020.

[8] B. Paaßen, J. Jensen, and B. Hammer. Execution traces as a powerful data representation for intelligent tutoring systems for programming. *International Educational Data Mining Society*, 2016.

[9] A. Patil. Automatic grading of programming assignments. 2010.

[10] T. W. Price, Y. Dong, and T. Barnes. Generating data-driven hints for open-ended programming. *International Educational Data Mining Society*, 2016.

[11] Y. Shi, K. Shah, W. Wang, S. Marwan, P. Penmetsa, and T. Price. Toward semi-automatic misconception discovery using code embeddings. In *The 11th International Conference on Learning Analytics and Knowledge (LAK 21)*, 2021.

[12] V. J. Shute. Focus on Formative Feedback. *Review of Educational Research*, 78(1):153–189, 2008.

[13] A. Stahlbauer, C. Frädrich, and G. Fraser. Verified from scratch: Program analysis for learners' programs. In *35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020*, pages 150–162. IEEE, 2020.

[14] A. Stahlbauer, M. Kreis, and G. Fraser. Testing scratch programs automatically. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 165–175, 2019.

[15] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

[16] W. Wang, A. Kwatra, J. Skripchuk, N. Gomes, A. Milliken, C. Martens, T. Barnes, and T. Price. Novices' learning barriers when using code examples in open-ended programming. ITiCSE '21. Association for Computing Machinery, 2021.

[17] W. Wang, Y. Rao, Y. Shi, A. Milliken, C. Martens, T. Barnes, and T. W. Price. Comparing feature engineering approaches to predict complex programming behaviors. 2020.

[18] W. Wang, Y. Rao, R. Zhi, S. Marwan, G. Gao, and T. W. Price. Step tutor: Supporting students through step-by-step example-based feedback. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pages 391–397, 2020.

[19] W. Wang, C. Zhang, A. Stahlbauer, G. Fraser, and T. Price. Snapcheck: Automated testing for snap programs. ITiCSE '21. Association for Computing Machinery, 2021.

[20] W. Wang, R. Zhi, A. Milliken, N. Lytle, and T. W. Price. Crescendo : Engaging Students to Self-Paced Programming Practices. In *Proceedings of the ACM Technical Symposium on Computer Science Education*, 2020.

[21] M. Wu, M. Mosse, N. Goodman, and C. Piech. Zero shot learning for code education: Rubric sampling with deep learning inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 782–790, 2019.

[22] R. Zhi, S. Marwan, Y. Dong, N. Lytle, T. W. Price, and T. Barnes. Toward Data-Driven Example Feedback for Novice Programming. *Proceedings of the International Conference on Educational Data Mining*, pages 218–227, 2019.

Is It Fair? Automated Open Response Grading

John A Erickson
Worcester Polytechnic Institute
jaerickson@wpi.edu

Anthony F Botelho
Worcester Polytechnic Institute
abotelho@wpi.edu

Zonglin Peng
Independent Researcher
zpeng@wpi.edu

Rui Huang
Independent Researcher
rhuang2@wpi.edu

Meghana V. Kasal
Independent Researcher
mkasalvinayakuma@wpi.edu

Neil T Heffernan
Worcester Polytechnic Institute
nth@wpi.edu

ABSTRACT

Online education technologies, such as intelligent tutoring systems, have garnered popularity for their automation. Whether it be automated support systems for teachers (grading, feedback, summary statistics, etc.) or support systems for students (hints, common wrong answer messages, scaffolding), these systems have built a well rounded support system for both students and teachers alike. The automation of these online educational technologies, such as intelligent tutoring systems, have often been limited to questions with well structured answers such as multiple choice or fill in the blank. Recently, these systems have begun adopting support for a more diverse set of question types. More specifically, open response questions. A common tool for developing automated open response tools, such as automated grading or automated feedback, are pre-trained word embeddings. Recent studies have shown that there is an underlying bias within the text these were trained on. This research aims to identify what level of unfairness may lie within machine learned algorithms which utilize pre-trained word embeddings. We attempt to identify if our ability to predict scores for open response questions vary for different groups of student answers. For instance, whether a student who uses fractions as opposed to decimals. By performing a simulated study, we are able to identify the potential unfairness within our machine learned models with pre-trained word embeddings.

Keywords

Natural Language Processing, Unfairness, Deep Learning, Word Embeddings, Pre-Trained Word Embeddings, Simulated Study

1. INTRODUCTION

In recent years, natural language processing (NLP) has been at the forefront of machine learning in multiple fields. Linguistics provides another source of information outside the standard data from user logs. Instead of relying on correlational assumptions from this data, inferences can be deduced directly from the users linguistics. While utilizing linguistics in education isn't genuine, modern machine learning and natural language processing has helped to automate the analysis and provides effective tools for learning.

The development of more advanced deep learning has brought a deeper semantic understanding of words within these linguistical models. The emergence of word embeddings were an important development in machine learning and NLP, but the publishing of publicly available pre-trained word embeddings, such as such as GloVe [21] or Wikipedia or Word2Vec [19], provided researchers with a powerful tool for optimizing algorithms with linguistics. While word embeddings were powerful for studies within areas such as MOOCS (i.e [14] [20]), smaller studies, with less robust linguistic data, were unable to utilize this modern approach for semantic relationship of words.

Since research has shown that some of the semantic meanings inferred from pre-trained word embeddings can elicit undesirable biases [2], the major question then becomes, does this underlying bias suggest the algorithm or predictive model will make unfair decisions? For instance, if an algorithm utilizes linguistics and NLP with pre-trained word embeddings will the predictions be unfairly made from those underlying biases. Our research attempts to explore:

1. Whether, through 3 simulated studies, the format a student writes an answer (i.e. fractions vs. decimals) effect the scoring model and potentially elicit unfair scoring?
2. What effect, through 3 simulated studies, if any, do *'distractor'* words have on the unfairness?
3. Whether or not underlying bias in pre-trained word embeddings can lead to unfairness in open response scoring models in middle school mathematics?

2. BACKGROUND

2.1 Intelligent Tutoring Systems

In recent years, online educational technologies have been on the forefront of learning for students. A common online educational technology, intelligent tutoring systems (ITS) [4],

John A. Erickson, Anthony F. Botelho, Zonglin Peng, Rui Huang, Meghana V. Kasal and Neil Heffernan "Is It Fair? Automated Open Response Grading". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 682-687. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

has been prevalent in education for many years. Some of the most common ITS are ASSISTments[11], McGraw Hill's ALEKS^{TM} and/or Carnegie Learning's Cognitive Tutor^{TM}. Through the use of both machine learning and software engineering, these systems have been shown to be effective at increasing the scores of students with end of the year standardized math exams[25] and the effects of their intelligent tutoring closely resembles the effect face to face tutoring has on students[31]. Other ITS, such as AutoTutor[9],have attempted to resemble the face to face tutoring more directly by developing automated conversations and dialogues between students and ITS [9]. However, most of the support and benefits of these ITS have been limited to questions with structured answers (i.e. multiple choice or fill in the blank questions).

2.2 Automation of Intelligent Tutoring Systems

Automated support of ITS is a draw for many teachers; one study noted that many utilize multiple choice questions for the efficiency and accuracy of grading [26]. However, since most of the automation is limited to questions with structured answers, the content which teachers provide is limited. Studies have looked to utilize NLP to automatically evaluate work or questions which require a student's unique linguistics (i.e. open response questions, or essays) including [29][28][24][1][7][30][17]. While most of this research has been primarily focused on content outside of mathematics, our previous research, [6], looked to help teachers diversify the content which they provide students in middle school mathematics by utilizing traditional and modern NLP to develop an automated scoring model for open response middle school mathematics questions. A more diverse set of question types can be beneficial to students and can elicit differing levels of cognition, as studies [18][15] note.

2.3 Natural Language Processing

Towards the goal of automating open response questions, or any linguistical/NLP prediction task, the major task is in how to numerically represent words thus that a machine learned algorithm can generate an accurate prediction. One of the more simplistic approaches utilizes the frequencies of each unique word within the corpus, whats commonly known as a *Bag of Words* approach. While undoubtedly easy to interpret and not computationally intensive, this approach has been utilized in studies such as [13] and is the foundation of more advanced approaches such as[27][10][22]. This has evolved into utilizing deep learning to generate word embeddings such as GloVe[21] and Word2Vec[19].

2.4 Pre-Trained Models

Embeddings are only as powerful as the data their train on. Not all researchers have robust corpuses, thus embeddings can be misleading. Pre-trained embeddings, such as GloVe or Word2Vec, publish their own embeddings generated from Wikipedia and GoogleNews. As these pre-trained word embeddings have grown in popularity, word embeddings have expanded to utilize bidirectional encoder representations from transformers (referred to as BERT[5]) to create pre-trained word embeddings, as well. Similarly, this has evolved from word level embeddings to pre-trained sentence and document level embeddings[23][3][16].

2.5 Fairness

When it comes to linguistics, the way someone speaks, the way someone articulates can be unique to themselves. Similarly, the way someone writes is personal to themselves and specific to their topic. So when algorithms are being pre-trained on data which isn't the researchers own data, there are questions to be asked. Research[2], has been able to identify some potentially harmful semantic relationships present in common pre-trained word embeddings. For instance, [2] was able to identify that Google's pre-trained Word2Vec on GoogleNews elicited some harmful stereotypes. There in lies the important question, if we omit variables which could cause unfairness in the automated scoring, are we continuing to avoid unfairness if we utilize pre-trained word embeddings. To identify this, we utilized Absolute Between-ROC Area (ABROCA) [8].

3. STUDY 1: SIMULATION STUDY

This research developed a simulated study to attempt to identify if pre-trained word embeddings are utilized within an automated scoring model for open response answers, do they influence the model to make unfair predictions. An example of this would be if a group of students state their answer with a fraction and surrounding text, does the predictive model generate scores similarly for those students that use decimals along with surrounding text? Through this simulated study, we are able to gain a deeper insight into what/if any unfair scoring occurs when utilizing the pre-trained GloVe word embeddings trained on Wikipedia between groups.

There are 3 studies within this simulated study to help achieve this goal. First, we develop answers which contain differing distributions of answers which contain fractions and decimals and generate the ABROCA value at the differing distributions. Second, we attempt to see if decimals and fractions alone generate differing ABROCA values. Third, we attempt to see if we replace decimals in the text with unknown tokens (more reliance on distractor words), do the ABROCA values differ at differing distributions? These studies will help provide deeper insight into the potential unfairness an automated scoring model can be producing when utilizing pre-trained word embeddings

3.1 Data Generation

At the foundation of this simulated study is the generation of the *student* dataset.The generation was split into two facets, the training dataset student answers and the test set student answers. This was performed such that the model would not be able to have any identical answers between the training set and the test set. Essentially, that the predictions aren't being made because the model has already seen that exact series of embeddings associated with a certain score.

Towards simulating authentic student answers, the generation of the corpus was founded on the goal of utilizing random selection. For the training set, as Table 2 shows (see Appendix A), there are 4 different length student answers in this corpus. There are answers which are 6, 5, 4 and 3 word length answers. The generation of the student answers can be surmised into 4 steps and visualized with Table 2 (see Appendix A). First, select whether it will be a student answer which uses decimals or fractions. Then,

randomly select what length the answer is. Once a length is randomly selected, another random selection is made between the two structures (i.e. 'Structure' within Table 2 in Appendix A). Finally, randomly select text from **Fill "1"** and **Fill "2" Fractions** or **Fill "2" Decimal** to fill the identifiers **'1'** and **'2'**.

This is the same approach that is utilized within the test set corpus generation as well. Table 3 (see Appendix A) shows that however there are different structures and phrases to select from than the training. Allowing for variability between test and training; thus, guaranteeing that a answer structure used in training isn't the same as in the test. From this, we can ascertain the model's predictions were not only based on identical phrases it has seen.

3.2 Methodology

This study sets out to identify if an automated scoring model for open response questions, which utilizes pre-trained word embeddings, elicit unfair scoring. With the answers generated, we set out to sample these simulated answers such that there is a balance of student answers which utilize fractions and decimals. The training set is comprised of student answers drawn from the pool of simulated answers containing fractions (considered as *Group A*), as well as answers sampled from similar student answers containing fractions and decimals as determined by a defined proportion threshold (considered as *Group B*).

A threshold was set for selecting decimals and fractions to control the balance of answers. This lends itself to our goal of being able to identify whether or not the format a student writes an answer, i.e. using factions vs. decimals, effects our ability to score student open response answers. Thus, with ABROCA, fairness can be identified at each threshold.

For the test set, a similar approach is taken. So the training and test set have both Group A answers which are distractor words and fractions, and Group B answers which have distractor words and a proportion of fractions/decimals (based on the same threshold for both training and test data).

To improve the reliability of the results, we re-sample/re-select the test dataset 10 times and evaluate the model's ability to score an open response answer. This form of cross validation allows us to see if the ability to predict the score was only for that unique set of words, or was the performance consistent across multiple iterations.

All of the studies will incorporate a Long Short Term Memory (LSTM) [12] model which utilizes the pre-trained word embeddings to automatically score open response answers and ABROCA is calculated. This is then used to run 3 studies. First, when incrementally increasing decimals being used in Group B, does the LSTM scoring model become more unfair? Second, whether or not fractions or decimals are the culprit of the potential unfairness in the automated scoring model by having answers just be fractions or decimals without distractor words? Third, when incrementally increasing unrecognized words (referred to as gibberish) being used in Group B, does the LSTM scoring model become more unfair? So does an imbalance in recognized words cause more unfairness between groups?

3.3 Results

In the end, Figure 1a showed that with the simulated study, when there is an increase in the proportion of decimals in Group B, there does not appear to be unfairness in the way Group A and Group B are evaluated. This is evident from the scale of the y-axis of Figure 1a, the ABROCA falls betweem 0.02 and around 0.04.

The second study, which there were only decimals and fractions in the test set (no distractor words), stayed constant at a ABROCA value of 0. The model was not unfair, it was able to equally evaluate both groups even when just isolated fractions and decimals.

The final simulation study, managed to show that increasing the imbalance between recognized and unrecognized tokens between groups increased the unfairness (ABROCA near 0.18). Figure 1b shows that the ABROCA score does indeed increase with more unrecognizable words within GloVe's pre-trained word embeddings. It should be noted that Table 1 shows some of the phrases used in the generated student answers were commonly associated with more correct answers

Table 1: Sample of Phrases and Their Associated Avg. Score

Generated Phrases	Avg. Score
my answer is	0.718750
i picked	0.622222
i guess the answer is	0.600000
i think it is	0.600000
i think the answer is	0.590909
i worked out	0.585366

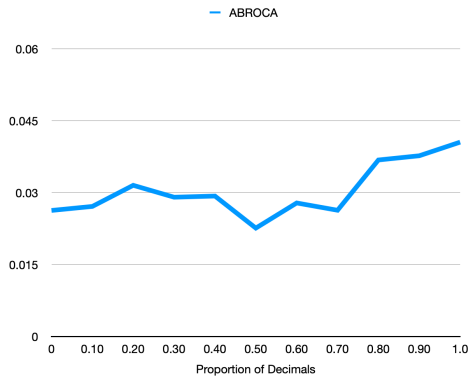
In the end, these simulated studies proved the largest risk for unfairness exists when there is differential coverage of answer-related tokens within applied methods utilizing pre-trained NLP embedding methods. So when answers consist of equally recognizable words within GloVe's pre-trained word embeddings, there's unlikely to be unfairness in the grading. There wasn't evidence that the inherent bias built into the pre-trained word embeddings elicited more unfair scoring of student answers in, in terms of this simulated study. But if there are unbalanced recognizable words and tokens in the student answers, attention needs to be paid to potential unfairness in the automated scoring.

4. STUDY 2: FAIRNESS IN REAL CONTEXTS

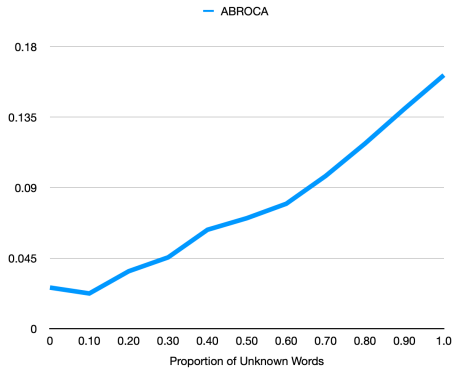
While a simulation study is powerful on its own, it is difficult to recreate authentic student data. For the final overall study of this research, we look to once again utilize ABROCA to identify if our own algorithm, trained on genuine student open response answers within ASSISTments, is unfair in its grading of women and men.

4.1 Data

The data consists of 141,612 graded authentic student open response answers across 2,042 unique problems. There were a total of 25,069 unique students who answered and 891 teachers graded those answers. Lastly, the scoring. This was performed on a 5 point scale, where students receiving a 4 is a perfect score.



(a) Study 1: ABROCA Values at Incremental Fraction/Decimal Thresholds



(b) Study 3: ABROCA Values at Incremental Fraction/Unknown Words Thresholds

It should be noted, to be able to perform the fairness analysis using ABROCA, gender was inferred. This performed by cross checking names with the census data. If the name was found only on the women or only on the men's list, it was labeled as such. In any names fell into multiple genders, it was labeled as unknown and excluded from this analysis.

4.2 Methodology and Results

Towards developing our predictions, we utilized another pre-trained algorithm, mentioned earlier, called SBERT. This is a pre-trained sentence embedding algorithm which allowed us to generate a single vector representation of each student answer. We then utilize a Canberra distance to identify which student answers are the most similar. Whichever was the most similar, that was the score we would assign. This approach managed to out do our previous models [6].

While utilizing, once again, ABROCA to identify potential unfairness, we apply this to our algorithm. We were able to show that our SBERT model with Canberra distance manages to fairly score both Male and Female student open response answers. Our model managed an ABROCA of 0.007, which is quite small, suggesting that our algorithm is indeed scoring fairly across these groups.

5. LIMITATIONS AND FUTURE WORK

While there were indications of unfairness in cases where there were unbalanced identifiable tokens within the student open response answers, this analysis is strictly middle school mathematics. This type of analysis would need to be applied to additional datasets to get a broader understanding of the potential unfairness in other subjects and age ranges. In terms of our analysis of our SBERT model for scoring student open response answers, while there wasn't unfairness identified, more work needs to be done to explore the embeddings themselves. Pre-trained word embeddings have been shown to have bias built in, but what bias is present in the pre-trained sentence embeddings? This is a question we look to explore further.

6. CONCLUSION

Overall, this study set out to run a simulated study to help identify potential unfairness within models utilizing pre-trained word embeddings. While there is bias present in the embeddings themselves, our simulated study didn't show this bias causing unfair scoring. However, our analysis did show that when developing models with pre-trained embeddings, unfairness can begin to occur when there is an imbalance of recognized tokens in the student answers. More specifically, our simulated study showed that when groups within the data use differing levels of recognized tokens, it increases the chance for unfair scoring.

While our simulated study showed how unfairness can present itself within a scoring model, our model on authentic student data did not show this unfairness. We were able to conduct an analysis of our model with ABROCA to compare our performance scoring identified male and female students.

In the end, we were able to utilize a simulated study to help identify potential unfairness in automated scoring models which utilize pre-trained word embeddings. Its been widely noted that those embeddings have bias built in, but our simulated study couldn't show an unfairness in the scoring of differing groups of simulated student answers. However, this study did suggest that there is a notable risk to fairness in cases where there are differences in the number of words that are unrecognized by pre-trained models across populations.

7. ACKNOWLEDGMENTS

We thank multiple NSF grants (e.g., 1917808, 1931523, 1940236, 1917713, 1903304, 1822830, 1759229, 1724889, 1636782, 1535428, 1440753, 1316736, 1252297, 1109483, & DRL-1031398), as well as the US Department of Education for three different funding lines; the Institute for Education Sciences (e.g., IES R305A170137, R305A170243, R305A180401, R305A120125, R305A180401, & R305C100024), the Graduate Assistance in Areas of National Need program (e.g., P200A180088 & P200A150306), and the EIR. We also thank the Office of Naval Research (N00014-18-1-2768) and finally Schmidt Futures we well as a second anonymous philanthropy.

8. REFERENCES

[1] Y. Attali and J. Burstein. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3), 2006.

[2] T. Bolukbasi, K.-W. Chang, J. Zou, V. Saligrama, and

A. Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *arXiv preprint arXiv:1607.06520*, 2016.

[3] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[4] A. T. Corbett, K. R. Koedinger, and J. R. Anderson. Intelligent tutoring systems. In *Handbook of human-computer interaction*, pages 849–874. Elsevier, 1997.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] J. A. Erickson, A. F. Botelho, S. McAteer, A. Varatharaj, and N. T. Heffernan. The automated grading of student open responses in mathematics. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 615–624, 2020.

[7] P. W. Foltz, D. Laham, and T. K. Landauer. Automated essay scoring: Applications to educational technology. In *EdMedia+ innovate learning*, pages 939–944. Association for the Advancement of Computing in Education (AACE), 1999.

[8] J. Gardner, C. Brooks, and R. Baker. Evaluating the fairness of predictive student models through slicing analysis. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, pages 225–234, 2019.

[9] A. C. Graesser, K. VanLehn, C. P. Rosé, P. W. Jordan, and D. Harter. Intelligent tutoring systems with conversational dialogue. *AI magazine*, 22(4):39–39, 2001.

[10] A. C. Graesser, P. Wiemer-Hastings, K. Wiemer-Hastings, D. Harter, T. R. G. Tutoring Research Group, and N. Person. Using latent semantic analysis to evaluate the contributions of students in autotutor. *Interactive learning environments*, 8(2):129–147, 2000.

[11] N. T. Heffernan and C. L. Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4):470–497, 2014.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[14] Z. Kastrati, A. S. Imran, and A. Kurti. Weakly supervised framework for aspect-based sentiment analysis on students' reviews of moocs. *IEEE Access*, 8:106799–106810, 2020.

[15] K. Y. Ku. Assessing students' critical thinking performance: Urging for measurements using multi-response format. *Thinking skills and creativity*, 4(1):70–76, 2009.

[16] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[17] J. Liu, Y. Xu, and Y. Zhu. Automated essay scoring based on two-stage learning. *arXiv preprint arXiv:1901.07744*, 2019.

[18] M. E. Martinez. Cognition and the question of test item format. *Educational Psychologist*, 34(4):207–218, 1999.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[20] A. Onan and M. A. Toçoğlu. Weighted word embeddings and clustering-based identification of question topics in mooc discussion forum posts. *Computer Applications in Engineering Education*, 2020.

[21] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[22] J. Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

[23] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[24] B. Riordan, A. Horbach, A. Cahill, T. Zesch, and C. M. Lee. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–168, 2017.

[25] J. Roschelle, M. Feng, R. F. Murphy, and C. A. Mason. Online mathematics homework increases student achievement. *AERA Open*, 2(4):2332858416673968, 2016.

[26] M. G. Simkin and W. L. Kuechler. Multiple-choice tests and student understanding: What is the connection? *Decision Sciences Journal of Innovative Education*, 3(1):73–98, 2005.

[27] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*, 2015.

[28] J. Z. Sukkarieh and J. Blackmore. c-rater: Automatic content scoring for short constructed responses. In *Twenty-Second International FLAIRS Conference*, 2009.

[29] J. Z. Sukkarieh, S. G. Pulman, and N. Raikes. Automarking: using computational linguistics to score short ,free- text responses. 2003.

[30] K. Taghipour and H. T. Ng. A neural approach to automated essay scoring. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1882–1891, 2016.

[31] K. VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221, 2011.

APPENDIX

A. SIMULATED ANSWER STRUCTURES

Table 2: Training Set Corpus Generation

Length	Structure	Answer Content	Fill "1"	Fill "2" - Fractions	Fill "2" - Decimals
6	6 - A	i **1** the answer is **2**	'think', 'believe', 'feel', 'suppose', 'guess'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
6	6 - B	**1** the answer **2**	'i arrived at', ' i worked out', 'ended up with'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
5	5 - A	i **1** it is **2**	'think', 'believe', 'feel', 'suppose', 'guess'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
5	5 - B	i **1** the answer **2**	'chose', 'thought', 'picked	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
4	4 - A	**1 2**	'i arrived at', ' i worked out', 'ended up with'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
4	4 - B	my **1 2**	'guess was', 'answer was', 'belief was', 'answer is'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
3	3 - A	i **1 2**	'chose', 'thought', 'picked	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
3	3 - B	it **1 2**	'was', 'is'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6

Table 3: Test Set Corpus Generation

Answer Length	Answer Structure	Answer Content	Fill "1"	Fill "2" - Fractions	Fill "2" - Decimals
6	6 - A	i 1 2	'arrived at the answer', 'thought the answer was' 'calculated the answer was', 'guessed the answer was'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
6	6 - B	2 1	'is the answer i thought', 'was the correct choice here', 'is what i guessed right', 'was what I arrived at'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
5	5 - A	1 2	'im guessing it was', 'my work arrived at', 'the answer is clearly', 'clearly the answer is'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
5	5 - B	2 1	'was what i guessed', 'is what i calculated' 'was the clear answer', 'is the correct answer'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
4	4 - A	1 2	'my guess is', 'my answer is', 'my work showed', 'my thought is'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
4	4 - B	2 1	'is my choice', 'from my work', 'is my answer', 'is the answer'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
3	3 - A	2 1	'is right', 'is correct', 'i found', 'i thought'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6
3	3 - B	1 2	'answer is', 'choice was', 'i guessed', 'i thought'	3/4, 1/2, 1/3, 2/5, 3/5	0.75, 0.5, 0.33, 0.4, 0.6

Predicting Executive Functions in a Learning Game: Accuracy and Reaction Time

Jing Zhang[1], Teresa Ober[2], Yang Jiang[3], Jan Plass[1] and Bruce Homer[4]
[1]CREATE Lab, New York University, New York, NY 10012
[2]LAMBS Lab, University of Notre Dame, Notre Dame, IN 46556
[3]Educational Testing Service, Princeton, NJ 08540
[4]The Graduate Center, City University of New York, New York, NY 10016
jz1220@nyu.edu, tober@nd.edu, yjiang002@ets.org, jp79@nyu.edu, BHomer@gc.cuny.edu

ABSTRACT
Executive functions (EF) are a set of psychological constructs defined as goal-directed cognitive processes. Traditional EF tests are reliable, but they are not able to detect EF in real-time. They cause a test effect if implemented multiple times. In contrast, learning games have the potential to obtain a real-time, unobtrusive measurement of EF. In this study, we analyzed log data collected from a game designed to train the EF sub-skill of shifting. We engineered theory-based game-level and level-specific features from log data. Using these features, we built prediction models with students' accuracy and reaction time during play to predict their standard measure of the EF shifting skill during the post-test and delayed post-test as well as to predict learning gains. Our model that predicts the post score has a correlation of 0.322 and that for the delayed post score is 0.303. The findings suggest that theory-based feature engineering and varying levels of granularity are two promising directions for cognitive skills prediction under the goal of game-based assessment. Also, accuracy, reaction time, and player progression are important features.

Keywords
Prediction, Game-Based Assessment, Learning Games, Executive Functions, Cognitive Skills.

1. INTRODUCTION
Executive functions (EF) are defined as "cognitive processes used for effortful, controlled, and goal-directed thinking and behavior" [29, 3, 4]. The unity/diversity model [24] views EF as consisting of related yet separable skills, which include updating, shifting (also termed cognitive flexibility), and inhibition. EF plays an important role in cognitive development and is associated with academic success [6], metacognitive skills [7], science learning [15], and language acquisition skills [10].

Game-based assessments allow educators to assess students' learning while they are playing a game and thus in a manner that can be highly efficient, fast, and entirely unobtrusive. Using games as assessments creates a context in which learners are likely to be highly engaged, which may optimally reflect their abilities [16, 28]. Using log data from digital games to evaluate learning is sometimes referred to as a "stealth assessment" [20] and has been used in the past decade to assess complex skills, such as creativity [33] and problem-solving, [34] based on log data. Log data collected during gameplay provides a record of student behaviors associated with EF and can be used for the prediction of EF [25]; however, is it possible to use log data collected from a game designed to train EF to measure EF *and* to develop a framework for game-based assessment?

Past studies of game-based assessments have focused on complex thinking skills, such as problem-solving [34]; however, there are constraints of game-based assessments of EF. First, it is necessary to determine the granularity or time scale for which we can detect students' EF in log data. Second, we need to separate log data related to EF training from log data related to other aspects of play to achieve a high performance of models. Third, we need to generate theory-based features relevant to EF skills. Accuracy and reaction time have been identified as indicators of EF [8].

This paper aims to provide proof of the concept for game-based assessments of the EF sub-skill of shifting. Shifting is one dimension of EF defined as the ability to switch attention between different "tasks, operations, or mental sets" [21]. The research questions include:

1. How do students' gameplay data predict their executive functions during a post-test and a delayed post-test?
2. Which features, including accuracy or reaction time, are important for predicting EF in games?

2. RELATED WORK

2.1 Games for EF Training and Measurement
Sustained and active engagement is widely thought to be critical for cognitive skills training games to be effective [2]. Incorporating gamified design features is one well-established mechanism for promoting meaningful engagement [9].

Digital training games can not only enhance EF [22, 27] but can also be used as a reliable means for measuring EF and other cognitive skills. For example, past work has examined the design and validation of computerized tools for measuring working memory capacity [21]. Previous research has also examined the use of a digital game for the detection of executive functions validated by a task for medical purposes in older adults using computational modeling [12]. Further work is needed to validate game-based measures of cognitive skills [35], especially those that are sensitive enough to detect variations among neurotypical individuals and that are appropriate for child and adolescent populations.

Jing Zhang, Teresa Ober, Yang Jiang, Jan Plass and Bruce Homer "Predicting Executive Functions in a Learning Game: Accuracy and Reaction Time". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 688-693. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

2.2 Game-Based Assessment

In previous research, the analysis of log data as a means of a formative assessment has yielded promising findings [11] and has been used for predicting a variety of cognitive and behavioral constructs, including quitting [19], knowledge [1], computational thinking [30], persistence [26], and implicit learning [31].

Evidence-Centered Design (ECD) [23] has been used effectively to develop game-based assessments in contexts that teach specific knowledge domains [14]. According to ECD, an assessment framework should take these models into account:

- *Task model:* Which actions is the learner taking within the system?
- *Evidence model:* Which features (e.g., from log data) can be used as evidence of learner actions?
- *Competency model:* How are these features associated with a set of standards or criteria that demonstrate effective learning has taken place?

Accounting for these three ECD models is helpful for feature engineering and predictive modeling. In game-based learning contexts that teach knowledge and skills, connecting a task model to an evidence model should be relatively straightforward given that the log data provides a detailed record of the learner's action sequence. Unlike game-based assessments of knowledge domains, where standards are clearly defined and may be validated by an expert review of content, in cognitive skills training game-based contexts, further work must be done to align an evidence model with a competency model. Accuracy and reaction time have been identified as two major aspects of an EF measurement [24], with evidence suggesting they each contributes uniquely to EF performance among children [8] and adolescents [5]. Yet, the way to distinguish the nuanced forms of accuracy and reaction time at varying levels of granularity and the way to combine them for predictive modeling within a game are currently unclear.

3. DATASET

3.1 Game Design

All You Can E.T. (AYCET) [17] is a game that trains the EF sub-skill of shifting. Its early prototype, The Alien Game, has been shown to improve EF after 1.5 hours of play for high school students [16] and two hours for college students [27]. In the current study, we used the "hot" version of AYCET, a version that maximizes the playfulness of the game. As Figure 1 shows, a player is asked to feed aliens with the appropriate food based on a certain rule. The rule changes multiple times at each level, thereby requiring the player to shift. As the player progresses in the game, the rule becomes more complicated.

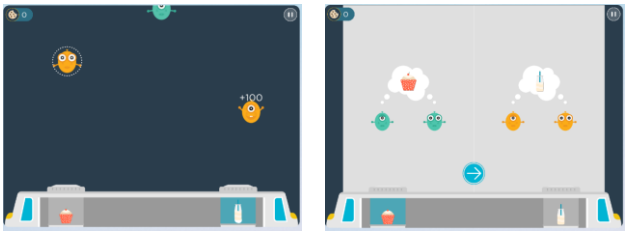


Figure 1. Feeding aliens and instructions for a rule.

3.2 Participants

Participants were recruited from three middle schools and two high schools in urban school districts in the Northeastern United States. They completed the study during non-instructional time at their schools. Among the 448 students who consented, 137 students were strategically randomly assigned to one of the three conditions to play AYCET throughout the study. Of those, 56 were removed because they demonstrated off-task behaviors, and thus the log data could not reflect their true ability. This resulted in an analytic sample of 81. Details of participant removal are discussed in the Data Cleaning subsection.

The 81 participants (M_{age} = 13.9 years, SD_{age} = 1.6, 46.1% female) included 39 in grade 7, 18 in grade 8, and 21 in grade 9. They reported a culturally and linguistically diverse background. Among them, 51.3% reported speaking Spanish at home, while 47.4% reported English and 1.3% Mandarin. As for ethnicity, 78.2% were Hispanic/Latino, 1.3% were Asian, 17.8% reported two or more ethnicities, 1.3% reported another ethnicity but did not specify, and 1.3% did not know. A few participants did not report their demographic information.

3.3 Study Procedure and Data Collection

The four-week intervention was conducted at the participating schools. Before gameplay, students completed a pretest. Then, they played the game for four sessions, each of which took about 30-40 minutes. The cumulative amount of time of play was 2-3 hours. After play, students completed a post-test, and an additional 4-8 weeks later, they completed a delayed post-test. The EF sub-skill of shifting was measured by the Dimensional Change Card Sorting (DCCS) task [36] in the pretest, post-test, and delayed post-test.

The log data consisted of 144,187 data points or actions, recording whether students fed each target ("alien") correctly or not and the reaction time for each target. This means that each alien required one action from the student. In this study, students played levels 1-30. There are 30-80 aliens per level.

In this study, each session began a few levels back from the last level played. After a few sessions, students were mandatorily pushed to level 11 to ensure they had enough time to play more difficult levels. This affected 72% of the students who were at level 9 or lower at the moment. On average, they were pushed by 4.3 levels.

3.4 DCCS Test and Score

The DCCS task [36] was used to measure the EF sub-skill of shifting in the pretest, post-test, and delayed post-test. Scoring was based on the National Institute for Health (NIH) scoring procedure [37]. This is a combination of the accuracy score and the reaction time score. The score ranges from 0 to 10. Floor or ceiling effects were not observed with our participants, as the top 25% of pretest scores ranged between 7.78 and 9.36.

4. METHOD

4.1 Data Cleaning

Based on the researchers' observations, we removed 33 participants for the following reasons: (1) did not complete one of the DCCS tests, (2) were off-task during the DCCS test, or (3) experienced technical difficulties that would affect their performance in the DCCS test. Furthermore, 23 participants were removed due to off-task behaviors (e.g., sleeping, non-stop talking, etc.) or an absence for at least one intervention session.

Eighty-one students remained in the analytic sample. The retention rate was 59.1%, which is acceptable for two reasons. First, data collected in the classroom setting are usually messier than that in the lab setting. During the study, a few participants went to the bathroom for a long time, which would be less likely to happen in a lab setting. Second, the game's focus on training EF required a

degree of attention that some students were not willing to invest. Some students found it difficult to remain attentive for an extended period of time.

4.2 Labels

Table 1 lists the labels for prediction. The post score and delayed post score were directly measured by the DCCS test. We next calculated the post-learning gain and delayed post-learning gain.

Table 1. Labels

Name	Description
post score	The EF score for the post-test.
post-learning gain	Relative gain of the EF score for the post-test compared with the pretest. Based on Hake's formula of learning gain [13], it is calculated as (post score - pre score)/(10 - pre score) because the EF score ranges from 0 to 10.
delayed post score	The EF score for the delayed post-test.
delayed post-learning gain	Relative gain of the EF score in the delayed post-test compared with the pretest.

4.3 Feature Engineering

We generated 20 game-level features and five level-specific features for each level that indicated student performance and progress. They capture information related to accuracy and reaction time in various mathematical formats and granularities.

Level-specific features were features for a single level. They included the average reaction time, the standard deviation of reaction time, accuracy, the number of correct hits (i.e., an action of feeding an alien with the correct food), and the number of wrong hits (i.e., an action of feeding an alien with the wrong food) across all aliens in a single level. Accuracy was calculated as the number of correct hits divided by the total number of aliens in a level.

Game-level features were aggregated features across all levels. They included: (1) the average, maximum, minimum, range, and standard deviation of a student's accuracy across all levels after calculating the accuracy for a single level across all aliens in that level; (2) the average, maximum, minimum, range, and standard deviation of a student's reaction time across all levels after taking the average reaction time for a single level across all aliens in that level; (3) the total number of correct hits (82% of all aliens among all students), wrong hits (16%), and missed hits (i.e., an action that the student did not feed an alien) (1%); (4) the highest number of stars a student received across all levels and the total number of stars a student received in the game; (5) the number of levels a student skipped by choice (which only happened before level 10) and due to the mandatory push; and (6) the highest level and the total number of levels a student played (as a student may skip a few levels).

4.4 Model Training

We used the linear regression for predictive modelling in RapidMiner 9.3. We evaluated the model's performance using ten-fold cross-validation at the student level to ensure the model would be generalizable to a new student population. During this process, students were randomly split into 10 groups. For each possible combination, we used forward selection to select features and then built the model based on the training data. Forward selection was an iterative process. First, a single-feature model that would achieve the highest Pearson correlation was chosen. Next, the remaining features were subsequently added one-by-one to the model if they could appreciably improve the model goodness of fit. In addition, to avoid collinearity, we set the minimum tolerance for eliminating collinear features as 0.05 and set "eliminate collinear features" as true in the linear regression operator.

In addition, we explored different combinations of features for feature selection. Missing values existed for many level-specific features. Thus, we began with the first feature set containing all game-level features and level-specific features of the level with the smallest missing value rate. After that, in each round, we added level-specific features of another level based on the ranking of the missing value rate. We stopped doing so at a level that contained missing data for 16% of students. The last model contained 65 features. In this way, we controlled for over-fitting and ensured the models were trained on representative levels.

5. RESULTS

5.1 Intervention Effect

The paired samples t-test show that the post score (mean = 6.98, SD = 1.56) is significantly higher than the pretest score (mean = 6.31, SD = 2.12) ($t(80) = 3.01$, $p < 0.01$, *Cohen's d* = 0.34). Also, the delayed post score (mean = 7.16, SD = 1.33) is significantly higher than the pretest score ($t(80) = 3.90$, $p < 0.001$, *Cohen's d* = 0.43). The boxplot of three EF scores is shown in Figure 2.

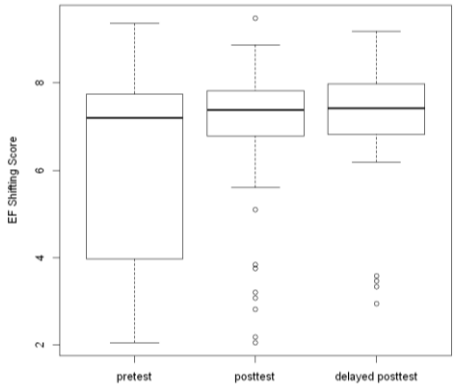


Figure 2. Boxplot of three EF shifting scores.

5.2 Correlation between Features and Labels

Among the 65 features for modeling training, five features had an absolute value of correlation with the post score between 0.3 to 0.42. Eight features had an absolute value of correlation with the delayed post score between 0.3 to 0.45. Features were weakly correlated with two learning gain labels as the absolute values of all correlation coefficients are below 0.25. Missing values for each feature were replaced with the average value of that feature.

5.3 Findings from Predictive Models

Cross-validated metrics for the best models of each label and their features are summarized in Table 2.

Table 2. Summary of the predictive models

	Post Score	Post-Learning Gain
RMSE	1.586	0.682
Correlation	0.322	0.294
Selected Features	- 2.087 * numLevelsSkippedByChoice - 0.008 * numWrongLevel11 - 0.028 * numWrongLevel12	0.414 * avgLevelAvgRT + 0.065 * numLevelsSkippedByPush - 0.001 * numCorrectLevel3 + 0.004 * numCorrectLevel4 + 0.493 * avgReactionTimeLevel2 - 0.428 * avgReactionTimeLevel13

Table 2 (continued). Summary of the predictive models

	Delayed Post Score	Delayed Post-Learning Gain
RMSE	1.540	0.470
Correlation	0.303	0.260
Selected Features	2.189 * avgLevelAvgRT - 0.268 * numLevelsSkippedByPush - 0.012 * numWrongLevel3 + 0.006 * numCorrectLevel12 - 2.154 * avgReactionTimeLevel3 + 1.693 * stdReactionTimeLevel3 - 2.306 * stdReactionTimeLevel12 - 0.528 * avgReactionTimeLevel12	0.841 * avgLevelAvgRT - 0.262 * highestLevelAvgRT + 0.001 * totalWrongHits + 1.434 * avgCorrectLevel1 - 0.003 * numWrongLevel3 + 0.007 * numCorrectLevel12 - 0.009 * numWrongLevel12 - 0.611 * stdReactionTimeLevel12

The models that only used game-level features had a low performance. Excluding level-specific features only did not greatly affect model goodness when predicting the post score, with a correlation of 0.308 and RMSE of 1.589. Features included *numLevelsSkippedByChoice*, *totalWrongHits*, and *numLevelsPlayed*.

6. DISCUSSION AND CONCLUSIONS

Playing AYCET significantly improved students' EF. The effect sizes of EF gains were medium, and that for the delayed post-test 4-6 weeks later was larger than that for the post-test. This difference in effect sizes may be attributed to either the long-term intervention effect by the EF game or students' natural development of EF. Though more evidence is needed, long-term effects of cognitive skills training have been found [18, 32].

We explored the possibility of a game-based assessment of EF using a game originally designed to train EF. We present four linear regression models that use the log data to predict students' EF score of shifting in the post-test, delayed post-test, and the relevant learning gain scores. With correlations around 0.3, these models achieved good performance for preliminary work. This corresponds to the second challenge of this study, which is to separate log data related to EF training from log data related to play in the game context. Good performance of predictive models indicates that a learning game is a promising tool to measure EF.

We generated an extensive list of game-level and level-specific features consisting of accuracy and reaction time indicators. Both accuracy and reaction time features are important in predicting EF but are two potentially distinct dimensions of EF. Generally, at the game level, a moderately higher reaction time and a more consistent reaction time (while controlling for other factors) are positively associated with EF. In addition, a lower reaction time and perhaps a more consistent reaction time are positively associated with EF. As for accuracy features, both correct hits and wrong hits are important for predicting EF.

In addition to accuracy and reaction time features, the number of levels skipped, particularly by the player, was indicative of EF. This means that player progression and player performance are both important for predicting EF.

Most selected features are from level 3 and level 12. This may suggest the key time window, which is the moment after students become familiar with the game mechanics and the moment after a drastic change in difficulty (recall the mandatory push; see section 3.3) may best demonstrate their ability to perform shifting. Varying the difficulty of levels or allowing for some time for students to achieve level 12 may contribute to a better game-based assessment of EF.

Responding to the first challenge, namely, the granularity and time scale for prediction, we found that level-specific features provide more promising results than game-level features only. It is worth further exploring variables at the action-level.

7. IMPLICATIONS AND FUTURE WORK

We explored the techniques of feature engineering and model training to investigate the game-based assessment of EF. The model performance is promising among studies that relate log data with a post-test measure in learning games [33, 34]. Another implication of this work is it sets the foundation for the real-time detection of EF and may provide the basis for dynamic interventions.

Limitations in the current work inspire us to explore more possibilities of game-based assessments for EF. First, a level may be played multiple times by a student. In the current study, all attempts of the same level were aggregated. In the future, we will distinguish multiple attempts of the same level by generating features such as the number of attempts and performance change over attempts. Second, we found that students' performance one or two levels after a challenging level is important. This game mechanic of difficulty change may not apply to other games. We have tried to interpret the model in the context of the specific game design. Third, students experienced a mandatory push in this study (see section 3.3). This is perhaps why features for level 12 were selected. To examine the generalizability of our findings, we will compare the prediction models built under two conditions, one of which replicates the push, while one does not; however, for practical reasons, it is also of interest to determine whether features that only cover earlier levels can predict the post-test and delayed post-test scores as this would require less game play for the assessment. Fourth, we filled the missing data with the average value of a feature. We did so by assuming data were missing at random. More robust methods, such as multiple imputations, could be used moving forward.

In the future, we are interested in generating theory-based features at the action-level (i.e., alien-level) per student, hoping to allow for the real-time detection of EF and for an even better model. An action-level feature may be a student's change in accuracy and reaction time within the first three aliens when the rule changes within a level. Another action-level feature may be the performance curve under different rules within a level. Both features align with the definition of the EF sub-skill of shifting and are not tied to specific levels, so they may produce more generalizable results. Methodologically, we are also interested in comparing the linear regression with other models, such as Support Vector Machines or the Random Forest. Substantively, it may be worth considering accuracy and reaction time as separate outcomes given research suggesting each contribute uniquely to performance on EF tasks in young children [9]. Further work may also apply methods of student modeling to other EF sub-skills, such as inhibition and updating in games that target these skills.

8. ACKNOWLEDGMENTS

Special thanks to Dr. Ryan Baker and Dr. Luc Paquette for their constructive suggestions for the data analysis. This work was supported in part by a research grant from the Institute of Educational Sciences, US Department of Education (Grant #R305A150417). The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education. We would like to thank the students and teachers who participated in this research.

9. REFERENCES

[1] Alonso-Fernández, C., Martínez-Ortiz, I., Caballero, R., Freire, M., and Fernández-Manjón, B. 2020. Predicting students' knowledge after playing a serious game based on learning analytics data: A case study. *Journal of Computer Assisted Learning*. 36, 3, 350-358.

[2] Anguera, J. A. and Gazzaley, A. 2015. Video games, cognitive exercises, and the enhancement of cognitive abilities. *Current Opinion in Behavioral Sciences*. 4, 160-165.

[3] Banich, M. T. 2009. Executive function: The search for an integrated account. *Current Directions in Psychological Science*. 18, 2, 89-94.

[4] Best, J. R. 2012. Exergaming immediately enhances children's executive function. *Developmental Psychology*. 28, 5, 1501-1510.

[5] Best, J. R. and Miller, P. H. 2010. A developmental perspective on executive function. *Child Development*. 81, 6, 1641-1660.

[6] Best, J. R., Nagamatsu, L. S., and Liu-Ambrose, T. 2014. Improvements to executive function during exercise training predict maintenance of physical activity over the following year. *Frontiers in Human Neuroscience*. 8, 353.

[7] Bryce, D., Whitebread, D., and Szűcs, D. 2015. The relationships among executive functions, metacognitive skills and educational achievement in 5 and 7 year-old children. *Metacognition and Learning*. 10, 2, 181-198.

[8] Camerota, M., Willoughby, M. T., Magnus, B. E., and Blair, C. B. 2020. Leveraging item accuracy and reaction time to improve measurement of child executive function ability. *Psychological Assessment*. 32, 12, 1118-1132.

[9] Cardoso-Leite, P., Joessel, A., and Bavelier, D. 2020. 18 Games for enhancing cognitive abilities. In Plass, J. L., Mayer, R. E., and Homer, B. D. ed. *Handbook of Game-Based Learning*. MIT Press, 437-468.

[10] Fuhs, M. W., Nesbitt, K. T., Farran, D. C., and Dong, N. 2014. Longitudinal associations between executive functioning and academic skills across content areas. *Developmental Psychology*. 50, 6, 1698-1709.

[11] Greiff, S., Wüstenberg, S., and Avvisati, F. 2015. Computer-generated log-file analyses as a window into students' minds? A showcase study based on the PISA 2012 assessment of problem solving. *Computers and Education*. 91, 92-105.

[12] Hagler, S., Jimison, H. B., and Pavel, M. 2014. Assessing executive function using a computer game: Computational modeling of cognitive processes. *IEEE Journal of Biomedical and Health Informatics*. 18, 4, 1442-1452.

[13] Hake, R. R. 1998. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*. 66, 1, 64-74.

[14] Henderson, N., Kumaran, V., Min, W., Mott, B., Wu, Z., Boulden, D., ... and Lester, J. 2020. Enhancing student competency models for game-based learning with a hybrid stealth assessment framework. In *Proceedings of the 13th International Conference on Educational Data Mining*. International Educational Data Mining Society, 92-103.

[15] Homer, B. D. and Plass, J. L. 2014. Level of interactivity and executive functions as predictors of learning in computer-based chemistry simulations. *Computers in Human Behavior*. 36, 365-375.

[16] Homer, B. D., Plass, J. L., Rafaele, C., Ober, T. M., and Ali, A. 2018. Improving high school students' executive functions through digital game play. *Computers and Education*. 117, 50-58.

[17] Homer, B. D., Plass, J. L., Rose, M. C., MacNamara, A. P., Pawar, S., and Ober, T. M. 2019. Activating adolescents' "hot" executive functions in a digital game to train cognitive skills: The effects of age and prior abilities. *Cognitive Development*. 49, 20–32.

[18] Jaeggi, S. M., Buschkuehl, M., Jonides, J., and Shah, P. 2011. Short-and long-term benefits of cognitive training. *Proceedings of the National Academy of Sciences*. 108(25), 10081-10086.

[19] Karumbaiah, S., Baker, R. S., & Shute, V. 2018. Predicting quitting in students playing a learning game. In *Proceedings of the 11th International Conference on Educational Data Mining*. International Educational Data Mining Society, 167-176.

[20] Ke, F. and Shute, V. 2015. Design of game-based stealth assessment and learning support. *Serious Games Analytics*. (2015), 301-318.

[21] Khenissi, M. A., Essalmi, F., Jemni, M., Chang, T. W., and Chen, N. S. 2016. Unobtrusive monitoring of learners' interactions with educational games for measuring their working memory capacity. *British Journal of Educational Technology*. 48, 2, 224-245.

[22] Mayer, R. E., Parong, J., and Bainbridge, K. 2019. Young adults learning executive function skills by playing focused video games. *Cognitive Development*. 49, 43-50.

[23] Mislevy, R. J., Steinberg, L. S., and Almond, R. G. 2003. focus article: on the structure of educational assessments. Measurement: Interdisciplinary Research and Perspectives. 1, 1, 3-62.

[24] Miyake, A., Friedman, N. P., Emerson, M. J., Witzki, A. H., Howerter, A., and Wager, T. D. 2000. The unity and diversity of executive functions and their contributions to complex "frontal lobe" tasks: A latent variable analysis. Cognitive Psychology. 41, 1, 49-100.

[25] Ober, T. M., Brenner, C. J., Olsen, A., Homer, B. D., and Plass, J. L. 2021. Detecting patterns of engagement in a digital cognitive skills training game. *Computers and Education*. 165, 104144.

[26] Owen, V. E., Roy, M. H., Thai, K. P., Burnett, V., Jacobs, D., Keylor, E., and Baker, R. S. 2019. Detecting wheel-spinning and productive persistence in educational games. In *Proceedings of the 12th International Conference on Educational Data Mining*. International Educational Data Mining Society, 378-383.

[27] Parong, J., Mayer, R. E., Fiorella, L., MacNamara, A., Homer, B. D., and Plass, J. L. 2017. Learning executive function skills by playing focused video games. *Contemporary Educational Psychology*. 51, 141-151.

[28] Plass, J. L., Homer, B. D., and Kinzer, C. K. 2015. Foundations of game-based learning. *Educational Psychologist*. 50, 4, 258-283.

[29] Plass, J. L., Homer, B. D., Pawar, S., Brenner, C., and MacNamara, A. P. 2019. The effect of adaptive difficulty adjustment on the effectiveness of a game to develop executive function skills for learners of different ages. *Cognitive Development*. 49, 56-67.

[30] Rowe, E., Almeda, M. V., Asbell-Clarke, J., Scruggs, R., Baker, R., Bardar, E., and Gasca, S. 2021. Assessing implicit computational thinking in zoombinis puzzle gameplay. *Computers in Human Behavior*. 120, 106707.

[31] Rowe, E., Asbell-Clarke, J., Baker, R. S., Eagle, M., Hicks, A. G., Barnes, T. M., ... and Edwards, T. 2017. Assessing implicit science learning in digital games. *Computers in Human Behavior*. 76, 617-630.

[32] Schwaighofer, M., Fischer, F., and Bühner, M. 2015. Does working memory training transfer? A meta-analysis including training conditions as moderators. *Educational Psychologist*. 50, 2, 138–166.

[33] Shute, V. J. and Rahimi, S. 2021. Stealth assessment of creativity in a physics video game. *Computers in Human Behavior*. 116, 106647.

[34] Shute, V. J., Wang, L., Greiff, S., Zhao, W., and Moore, G. 2016. Measuring problem solving skills via stealth assessment in an engaging video game. *Computers in Human Behavior*. 63, 106-117.

[35] Valladares-Rodríguez, S., Pérez-Rodríguez, R., Anido-Rifón, L., and Fernández-Iglesias, M. 2016. Trends on the application of serious games to neuropsychological evaluation: A scoping review. *Journal of Biomedical Informatics*. 64, 296-319.

[36] Zelazo, P. D. 2006. The dimensional change card sort (DCCS): A method of assessing executive function in children. *Nature Protocols*. 1, 1, 297-301.

[37] Zelazo, P. D., and Bauer, P. J. 2013. ed. *National Institutes of Health Toolbox Cognition Battery (NIH Toolbox CB): Validation for Children Between 3 and 15 Years*. Wiley, Hoboken, NJ.

Leveraging Survey and Motion Sensors Data to Promote Gender Inclusion in Makerspaces

Edwin Chng, Stephanie Yang, Gahyun Sung, Tyler Yoo, Bertrand Schneider
chng_weimingedwin@g.harvard.edu, szhang@g.harvard.edu,
gsung@g.harvard.edu, tyler_yoo@g.harvard.edu, bertrand_schneider@g.harvard.edu
Harvard University

ABSTRACT
Over the last decade makerspaces have become more popular and prevalent in formal and informal learning environments. A finding, however, is that makerspaces are often male-dominated, and females can feel a sense of intimidation in the space. Furthermore, maker-centered learning typically adopts an open-ended structure which makes it difficult to identify students who are struggling. In this paper, we explore the use of quantitative data from survey and motion sensors to potentially assist instructors in uncovering gender differences and promoting gender inclusion. Results suggest that there are different pathways for male and female students to thrive in makerspaces. Based on survey results, male students tend to have higher self-efficacy, resulting in more self-confidence in their abilities and more positive feelings. Findings from applying network analysis on the motion sensor data show that female students persevere more consistently and use empathy to form closer ties with peers for mutual support. These findings suggest that quantitative data could help raise instructors' awareness of gender differences and use that information to cater to the unique learning needs of each group of students. Overall, this work represents preliminary steps in instrumenting makerspaces to promote gender inclusion and support maker-centered learning.

Keywords
Interaction Analysis, Learning Analytics / Educational Data Mining, Social Network Analysis, Broadening Participation, Gender, Making and Makerspaces, Technology-enhanced learning

1. INTRODUCTION
While many authors have espoused the learning benefits of makerspaces [5], other researchers have recognized the inherent difficulties of supporting student learning in makerspaces' open-ended environment [13]. First, students are expected to solve problems independently in open-ended learning environments. Such independent work may lead to feelings of isolation, and instructors may not be aware that students are struggling. Second, the iterative nature of work in the makerspaces makes it difficult for instructors to continuously monitor students' progress. Without a clear feedback system, it is challenging for instructors to differentiate when students need instructional support.

Edwin Chng, Stephanie Yang, Gahyun Sung, Tyler Yoo and Bertrand Schneider "Leveraging Survey and Motion Sensors Data to Promote Gender Inclusion in Makerspaces". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 694-698. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

However, new sensing technology (such as motion tracking) offers an opportunity to address some of these challenges. The key benefit of using motion sensors is that they can be deployed to monitor students' learning in a continuous and unobtrusive manner. Therefore, we aim to examine how the use of quantitative data can help instructors overcome inherent challenges of assisting students in makerspaces.

For our scope, we examine how students from different genders interact in makerspaces [2;8;11], and we hope to promote gender inclusion in makerspaces. Eventually, we hope that the use of quantitative data can assist instructors in identifying the right form of support for each diverse group of students.

2. LITERATURE REVIEW
Makerspaces draw learners from a diversity of disciplines and provide multiple entry points to participation leading to "innovative combinations, juxtapositions, and uses of disciplinary knowledge and skill" [12]. However, makerspaces situated in formal learning environments are often male dominated [7]. Hence, it is an increasing priority for makerspaces to include women who are underrepresented in these communities. Central to this goal is understanding how women interact within makerspace courses. While some studies have not found gender to be a salient factor [2], other studies have shown that women often report feeling intimidated and excluded [8;11]. Most studies conducted in this area have also been qualitatively based profiles. Yet, for instructors to better support women in these spaces, more research must be conducted on gender differences in the cognitive, non-cognitive and affective domains and understand how these differences contribute to the outcomes of empowerment and community-building in maker-centered learning [5].

In this regard, the use of quantitative data from motion sensors could help provide alternative insights into gender differences. Researchers in the field of multimodal learning analytics have long explored the use of sensors to gather information on student learning because data can be collected in a sufficiently high frequency to draw rich inferences [3]. Since social interactions are an important part of makerspace projects, we focus this paper on capturing them using motion sensors. The successful utilization of motion sensors in capturing student interactions have been suggested by a couple of researchers [4;9]. One common thread in these previous works is the use of physical proximity as a rudimentary proxy for interaction. While being in close proximity is a necessary condition for interaction to occur, it is arguably not a sufficient condition. Therefore, in addition to the use of physical proximity as an indication for interactions, this study will layer on two other criteria (see Section 5.3). In essence, we hope that the use of quantitative data from motion sensors can paint a broader picture of women's experiences in makerspaces to improve instructor support and inclusivity.

3. CONTEXT OF STUDY

Quantitative data was collected from 14 female and 8 male students enrolled in a 15-week makerspace course (no students identified as non-binary). Kinect sensors were deployed 24/7 to gather skeletal joint data from students and survey tools were used weekly to assess students' learning experiences.

3.1 Course overview

The graduate-level makerspace course took place at a school of education in the northeastern part of the United States. With a focus on digital fabrication, the course aims to equip students with the necessary skills and knowledge to handle modern tools such as laser cutters. Each week, students are expected to work on a course assignment that typically involves the creation of a digital fabrication product for educational purposes. Depending on the requirements of the assignment, students could either work individually or collaborate. In addition to these weekly assignments, students also pair up to complete mid-term and final projects. While instructional support is available in the form of office hours and individual consultations, students largely work independently with minimal intervention from instructors.

Because of the open-ended nature of makerspaces, the course is designed with several scaffolds. Every week, the same cycle of design-prototype-create is adopted for each course assignment. In this manner, students continually receive opportunities to refine their skills. The presence of these weekly cycles also provides the research team with a natural unit of analysis and all quantitative data collected is aggregated at the week level.

3.2 Kinect Setup

Six Kinect v2 sensors were deployed in the makerspace to collect skeletal joint data. The sensors were positioned to achieve maximum coverage of the space (see Figure 1). When an individual's presence is detected in the Kinect's field of vision, the Kinect starts to record the x,y,z coordinates of the individual's head joint, left and right shoulder joints, left and right elbow joints, and left and right-hand joints. When there are multiple individuals present in the space, each Kinect sensor has the capability of recording up to 6 individuals at 30 Hz (i.e., 30 observations per second).

4. RESEARCH QUESTIONS

RQ1: What gender differences can be extracted from quantitative data collected from a makerspace?

RQ2: Which quantitative factors can account for students' development of a sense of empowerment and community spirit?

We examined students' sense of empowerment and community spirit in the second research question because these are key attributes of a maker mindset [5].

5. METHODS

In order to investigate how different genders work and interact in the makerspace, we constructed social networks from Kinect observations and derived network measures for each student (described in section 5.2 and 5.3). Additionally, we conducted weekly surveys of students to better understand their learning experiences (section 5.1). These surveys not only served as a triangulation measure for the Kinect observations, but also complemented the data by providing a more holistic description.

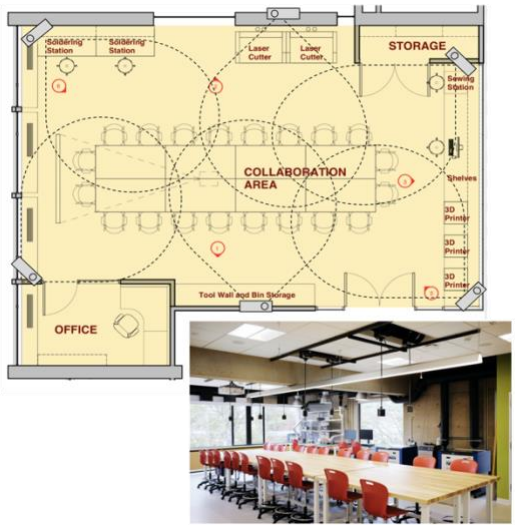


Figure 1. Layout of makerspace with positions and fields of vision of the Kinect sensors (top). Picture of the makerspace (bottom)

5.1 Survey Data

Surveys were administered to students after class each week. These surveys were crafted based on a literature review of surveys and to validate the questions, we conducted a validation study with students from a previous iteration of the course.

Table 1. Details of the surveys administered

Dimensions	Survey item	Scale	Source
Cognitive	- Tool Use - Time Committed	Likert 1-7 Numerical	General questions
Non-cognitive	- Perceived Competence - Self-Regulation - Motivation	Likert 1-5	[10; 15]
Affective	- Frustrated - Nervous - Interested - Inspired	Likert 1-5	[14]
Maker's attribute	- Sense of empowerment - Community spirit	Likert 1-5	[5]
Maker's mindset	- Can-do attitude - Empathy - Curiosity - Perseverance - Resourceful - Collaborative	Likert 1-7	[5]

Referencing Table 1, students' learning experiences were captured based on three dimensions: cognitive, non-cognitive, and affective. The two dimensions of maker's attribute and maker's mindset act as proxies for student outcomes. To determine gender differences, we conducted t-tests on these survey scores.

5.2 Kinect Data

Kinect observations were used for this study to infer the social interactions amongst students. Examining student interactions is key because communities represent an indispensable resource for students working in an open-ended environment. The following steps were taken to clean and process the Kinect data.

1) Student identification: Even though the Kinect sensors have no ability in establishing the identities of students, they capture video images from their fields of vision. These images were fed into OpenFace [1] to identify students.

2) Data homography: The coordinate system that the sensors operate in is relative to the actual positions of the sensors in the space. Hence, there is a need to convert the data into a coordinate system that better represents the 3D positions of the skeletal joints. Data homography was used to achieve this. A research team member stood in front of each sensor at nine different locations, forming a grid. Using the marked-out grid locations on the floorplan of the space and the measured positions of the skeletal joints, the coordinate system of sensors was translated into a coordinate system that is based on the floor plan (Figure 1).

3) Deduplication of skeletal joints: Finally, data from all six sensors were combined into a single coordinate system. However, because the sensors had overlapping fields of view, there was a possibility that multiple sets of skeletal joints were recorded for the same individual. In this case, deduplication was carried out to remove the additional skeletal joints for the same person.

5.3 Social Network Analysis

Once the Kinect data was processed, social networks were constructed. The social networks are built based on the episodes of student interaction. A student is said to have interacted with another if both students are one meter apart, have significant amounts of hand movement, and are either both sitting or both standing. The first criterion is based on the theory of proxemics [6], which states that humans maintain a comfortable distance of one meter during interactions. Admittedly, a proximity of one meter is a necessary but not sufficient criterion for establishing social interactions. Therefore, two other criteria were added to increase the probability of capturing true episodes of student interaction. The second criterion is based on the hands-on nature of the makerspace course. For two students who are in close proximity, having significant amounts of hand movement is likely an indication of collaboration. The third criterion is based on the observation that students tend to share the same eye level when working with each other. It is rare to observe two students working together with one individual standing and another sitting. While these three criteria are not perfect indicators of social interactions, observations of students working in the makerspace and crosschecks conducted by looking at screenshots from the sensors validated their use as a proxy for social interactions.

After we identified episodes of student interactions, social networks were generated based on the amount of time each student spent interacting with others. In essence, the nodes of the social network represent the individual students while the edges between nodes are weighted according to the amount of interaction time spent between students. From the weekly social networks, network measures were computed to obtain weekly network scores for each student.

Table 2. Details of network measures used

Network measures	Definition	Scale
Degree Centrality	This represents the fraction of nodes that a node is connected to.	0 to 1
Average edge weight	This is the mean of all the weights of all the edges connected to a node.	0 to inf
EI homophily index	This index is calculated by taking the difference between out-group and in-group connections and dividing by the total number of connections. For instance, in EI gender, a node for a female student would have male connections as out-group connections and female connections as in-group connections.	-1: Complete homophily (only in-group connections) 1: Complete heterophily (only out-group connections) 0: Equal number of in-group and out-group connections.

T-tests of the network measures were then conducted to extract gender differences, which addresses the first research question. For the second research question, the identified gender differences were used to build regression models for students' sense of empowerment and community spirit.

6. RESULTS

RQ1: What gender differences exist in a makerspace (from the quantitative measures)?

Table 3. Results of t-tests for gender differences

Measures	Statistical differences (t-test)
Survey: Perceived Competence	Males students (mean=5.2) reported having a higher level of perceived competence than female students (mean=4.8), t(20)=2.25, p=0.03.
Survey: Interested	Males students (mean=4.0) reported feeling more interested in the course than female students (mean=3.7), t(20)=2.21, p=0.03.
Survey: Inspired	Males students (mean=3.7) reported feeling more inspired in the course than female students (mean=3.4), t(20)=2.22, p=0.03.
Survey: Empowerment	Males students (mean=3.9) reported having a greater sense of empowerment than female students (mean=3.5), t(20)=3.11, p=0.002.
Survey: Empathy	Females students (mean=6.0) reported having more empathy than male students (mean=5.5), t(20)=2.14, p=0.04.
Survey: Perseverance	Females students (mean=5.9) reported having more perseverance than male students (mean=5.3), t(20)=2.71, p=0.008.
Network measure: EI gender	Males students (mean=0.02) have more diverse gender interactions than female students (mean=-0.16), t(20)=4.19, p<0.001.

Several items were different for males and females. First, males reported having higher perceived competence, which suggests that males are more confident individuals when it comes to assessing their abilities. Second, males recounted feeling more interested and inspired in the course. This shows that males possess more positive feelings towards the course. The lack of statistical significance for the negative affective states indicates that males and females might be struggling equally in the course. Third, males described developing a stronger sense of empowerment. This implies that males feel they have benefitted from the course and can move on to accomplish more challenging tasks.

Although males reported doing better in the course than females, the t-test results also indicate that females may possess some

alternate mechanisms for thriving in the course. Females score higher on empathy, which suggests females relate better to others in the community. Additionally, females score higher on perseverance, which hints at positive struggles from females. Lastly, for the network measures, females score more negatively in the EI index for gender, which implies that females interact more with other females, possibly for more community and emotional support.

RQ2: Which quantitative factors can account for students' development of a sense of empowerment and community spirit?

Findings from the survey data suggest that male and female students in this study differ in their perceived competence, positive feelings, empathy, perseverance and diversity in gender interactions. A linear regression model was built based on these to predict the students' sense of empowerment and community spirit.

Table 4. Regression models for sense of empowerment and community spirit (*p<0.05, **p<0.01, ***p<0.001)

Outcomes (→) Predictors (↓)	**Sense of empowerment****	**Community spirit*****
	F (4,17) = 6.85 p-value = 0.0018 R^2 = 0.6172 RMSE = 0.7915	F (1,20) = 16.72 p-value = 0.0006 R^2 = 0.4554 RMSE = 1.0513
const	coef. = -1.034 S.E. = 1.582 p-value = 0.522	coef. = -2.672 S.E. = 1.614 p-value = 0.113
Empathy		coef. = 1.123** S.E. = 0.275 p-value = 0.001
Perceived competence	coef. = 0.770 S.E. = 0.369 p-value = 0.052	
Positive feelings	coef. = 0.232* S.E. = 0.109 p-value = 0.048	
Perseverance	coef. = 0.954** S.E. = 0.242 p-value = 0.001	
EI gender	coef. = -0.441** S.E. = 0.111 p-value = 0.001	

Based on the regression analysis, students' positive feelings, perseverance, and diversity in gender interactions are significant predictors for their sense of empowerment. Even though perceived competence is not statistically significant, its low p-value of 0.052 hints that it might be a contributing factor to students' sense of empowerment (which corroborates with RQ1's findings). Similarly, the presence of positive feelings in the model echoes previous findings of males having more positive feelings and a greater sense of empowerment. However, it is unclear if students developed a greater sense of empowerment due to their positive feelings or if students felt more positive because they experienced empowerment. Lastly, the inclusion of perseverance and diversity in gender interactions as significant predictors demonstrates that initial learning difficulties in makerspaces can be overcome if one perseveres and that reaching out to fellow members for peer support can aid in the process of learning. Since females have expressed higher levels of perseverance and more in-group preferences previously, this finding reveals a potential pathway for female students to develop a sense of empowerment.

The regression analysis of community spirit shows that empathy is the sole significant predictor. Furthermore, the regression model with only empathy included has an R^2 value of 0.4554, which means that empathy as a factor alone can explain close to half of the variability in community spirit. This is not an unexpected finding as empathy remains a much-needed ingredient for the fostering of good relationships. This result also hints at possible contributions from females in building makerspace communities since they possess higher levels of empathy.

7. DISCUSSION

The findings of this paper indicate that males in this study are more confident in their technical ability and have more positive feelings associated with the makerspace. These findings run parallel to qualitative results in the literature which show that males tend to display more initial interest in makerspaces and technically oriented making activities [8]. While males self-reported more confidence in their abilities, females in this study were more persistent. Additionally, females reported higher measures of empathy and tended to interact more with other females when in the makerspace. These results are in line with qualitative findings from [11] indicating that females tend to appreciate having other females in the space.

In terms of promoting gender inclusion, the methods used in this study can help reveal to instructors the salient differences between genders operating in their own makerspaces. When awareness of gender differences is promoted, instructors can be naturally prompted to take more active steps to cater to distinct learning needs. Additionally, these findings serve as a reminder for instructors to avoid taking on a deficit view of any gender. On the surface, it might appear that males are thriving better than females in makerspaces, but the lack of statistical significance for the negative affective states signals that males and females struggle equally. Instead, our results suggest that males and females thrive in their unique ways in makerspaces, with males using their higher individual self-efficacy, and females using their greater group empathy skills. Neither males nor females should be viewed in a deficit perspective, and the removal of any gender bias would certainly go a long way in promoting gender inclusion.

Limitations of the current study include the relatively small sample size and the fact that the survey results were based on self-reported measures. These factors call into question the generalizability of our findings, and future work should seek to corroborate these results. Additionally, any reader of these findings must be careful to not fall into gender stereotypes. These results are reported on an aggregated basis, which may or may not be applicable to any individual student. Moreover, these findings are a result of our observations conducted in this particular study. Nonetheless, the findings demonstrate the feasibility of an approach that can be used by instructors to uncover gender differences in their own makerspaces.

8. CONCLUSION

The current paper examined gender differences in makerspaces and the factors that contributed to students' development of a sense of empowerment and community spirit. T-test results indicate that there are different pathways for male and female students to thrive in makerspaces and regression analyses highlight the quantitative factors that can account for students' development of a sense of empowerment and community spirit. This work presents preliminary steps in designing an automated system for instructional use to support gender inclusion.

9. REFERENCES

[1] Amos, B., Ludwiczuk, B., & Satyanarayanan, M. (2016). Openface: A general-purpose face recognition library with mobile applications. *CMU School of Computer Science, 6*(2).

[2] Bean, V., Farmer, N. M., & Kerr, B. A. (2015). An exploration of women's engagement in Makerspaces. *Gifted and Talented International*, 30(1–2), 61–67.

[3] Berland, M., Baker, R. S. & Blikstein, P. (2014). Educational data mining and learning analytics: Applications to constructionist research. *Technology, Knowledge and Learning*, *19*(1-2), 205-220.

[4] Chng, E., Seyam, M. R., Yao, W., & Schneider, B. (2020, July). Using Motion Sensors to Understand Collaborative Interactions in Digital Fabrication Labs. In *International Conference on Artificial Intelligence in Education* (pp. 118-128). Springer, Cham.

[5] Clapp, E. P., Ross, J., Ryan, J. O., Tishman, S. (2016). *Maker-centered learning: Empowering young people to shape their worlds.* Jossey-Bass, San Francisco.

[6] Hall, E. T. (1966). *The Hidden Dimension*. Anchor Books.

[7] Hynes, M. M., & Hynes, W. J. (2018). If you build it, will they come? Student preferences for Makerspace environments in higher education. *International Journal of Technology and Design Education, 28*(3), 867-883.

[8] Lewis, J. (2015). *Barriers to women's involvement in hackspaces and makerspaces*.

[9] Martinez-Maldonado, R., Yacef, K., Dos Santos, A. D. P., Shum, S. B., Echeverria, V., Santos, O. C., & Pechenizkiy, M. (2017). Towards proximity tracking and sensemaking for supporting teamwork and learning. In *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)* (pp. 89-91). IEEE.

[10] Pintrich, R. R., DeGroot, E. V. (1990). Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology, 82*, 33-40.

[11] Roldan, W., Hui, J. S., & Gerber, E. M. (2018). University makerspaces: Opportunities to support equitable participation for women in engineering. *International Journal of Engineering Education, 34*, 751-768.

[12] Sheridan, K., Halverson, E. R., Litts, B., Brahms, L., Jacobs-Priebe, L., & Owens, T. (2014). Learning in the Making: A Comparative Case Study of Three Makerspaces. *Harvard Educational Review*, 84(4), 505–531.

[13] Tan, M. (2019). When makerspaces meet school: Negotiating tensions between instruction and construction. *Journal of Science Education and Technology*, *28*(2), 75-89.

[14] Watson, D., Clark, L. A., Tellegan, A. (1988). Development and validation of brief measures of positive and negative affect: The PANAS scales. *Journal of Personality and Social Psychology, 54*(6), 1063-1070.

[15] Williams, G. C., Deci, E. L. (1996). Internalization of biopsychosocial values by medical students: A test of self-determination theory. *Journal of Personality and Social Psychology*, *70*, 767-779.

Early Detection of At-risk Students based on Knowledge Distillation RNN Models

Ryusuke Murata
Department of Advanced Information Technology
Kyushu University, Japan
murata@limu.ait.kyushu-u.ac.jp

Atsushi Shimada
Department of Advanced Information Technology
Kyushu University, Japan
atsushi@limu.ait.kyushu-u.ac.jp

Tsubasa Minematsu
Department of Advanced Information Technology
Kyushu University, Japan
minematsu@limu.ait.kyushu-u.ac.jp

ABSTRACT

Recurrent neural network (RNN) achieves state-of-the-art in several researches of the performance prediction. However, accuracy in early time steps is lower than that in late time steps, even though the early detection of at-risk students is important for timely interventions. To improve the accuracy in early time steps, we propose a knowledge distillation method for RNN. Our method distills the time-series information in the RNN model of late time steps into the RNN model of early time steps. This distillation makes the prediction of early time steps closer to that of late time steps. The experimental result showed that our method improved the detection rate of at-risk students compared with traditional RNNs, especially in early time steps.

Keywords

Student performance prediction, Early detection of at-risk students, Recurrent neural network, Knowledge distillation

1. INTRODUCTION

The detection of at-risk students is an essential task to ensure intervention as early as possible. At-risk students are those who may drop out of lecture courses and have low scores (e.g., grade point averages and quiz scores). When potential at-risk students are automatically detected in the early stage of courses, teachers can have sufficient time to encourage them to continue learning.

In recent years, prediction models based on recurrent neural networks (RNNs) have reached high performance [1, 3, 6, 7, 10, 11, 14, 19]. RNNs can handle time-series information such as weekly learning behavior and predict students' performance in each time step. Therefore, RNNs can detect at-risk students in each time step such as after each lecture. However, prediction accuracy in early time steps is lower than that in late time steps because it is difficult for RNNs to extract representative features from only the time-series information in early time steps.

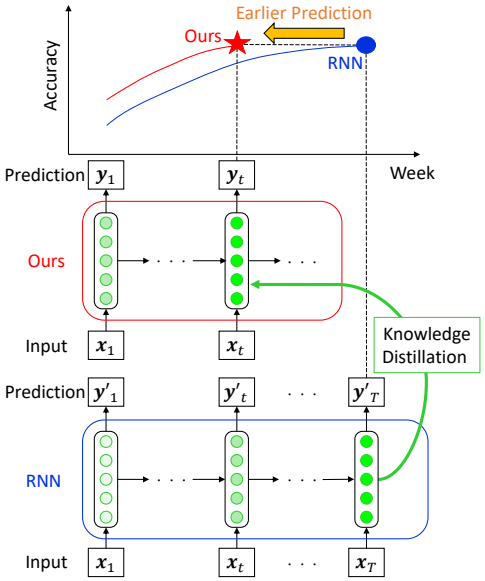


Figure 1: KD for the earlier detection of at-risk students.

To solve this problem, we propose a novel training strategy for improving the prediction in early time steps. Figure 1 shows an overview of our proposed method. Traditional RNNs can extract more representative features in later time steps, and prediction accuracy can also increase because RNNs can use longer time-series information. If RNNs can obtain more representative features from the inputs of earlier time steps, they can detect at-risk students earlier and maintain detection accuracy.

To transfer extracted features, we use *knowledge distillation* (KD) [4]. KD is a compression method for deep neural networks (DNNs), and many methods have been proposed in several fields such as visual recognition [2, 8] and natural language processing [5, 13, 15]. In KD, the model is compressed by training a small DNN model (student model) from a large DNN model (teacher model); that is, the knowledge in the teacher model is distilled to the student model. Further, KD does not require new annotations. In our method, KD is applied to transfer the representative features extracted from

Ryusuke Murata, Tsubasa Minematsu and Atsushi Shimada "Early Detection of At-risk Students based on Knowledge Distillation RNN Models". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 699-703. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

longer time-series information. As shown in Figure 1, this distillation makes the prediction of early time steps closer to that of late time steps, allowing us to detect at-risk students earlier.

The contributions of this study are summarized as follows.

- We introduce KD to predict students' performance. To the best of our knowledge, this is the first study to apply KD to performance prediction.
- We propose the RNN-FitNets model to improve early performance prediction. This model performs as if the learning behaviors in all the time steps are inputted, even though the model only receives the learning behavior in early time steps.
- We evaluate the effectiveness of our model for detecting at-risk students based on the learning logs collected from a higher education course.

2. KNOWLEDGE DISTILLATION RNN MODEL

In this study, we propose RNN-FitNets, which is an integration of RNNs and FitNets [12]. RNN-FitNets distils the well extracted features in the later time step into the RNNs in the earlier time steps by using the architecture of FitNets. Therefore, RNN-FitNets can improve the prediction accuracy in the earlier time steps. For example, as shown in Figure 1, RNN-FitNets can extract representative features in time step 3, whereas traditional RNNs obtain the same feature in time step T.

Figure 2 shows the architecture. The teacher model is pre-trained using all the time steps $(1, 2, ..., T)$, and the student model is trained until time step t $(1 \leq t \leq T)$. During the pre-training of the teacher model and training of the student model, the same ground truth holds (e.g., the final grade is passed to all the time steps). The teacher and student models have the same structure; only the time steps differ between them. Therefore, unlike FitNets, no regressor that transforms the size of the hidden layer of the student model exists.

The student model is trained using two steps in each training epoch as with FitNets. First, it updates its parameter, except for the output layer. Given the t-th time step feature vector of the student model as $\boldsymbol{h}_t$ and T-th time step feature vector of the teacher model as $\boldsymbol{h}'_T$, the parameter is updated by minimizing the following hint loss function L_{HT}:

$$L_{HT} = \frac{1}{2}\|\boldsymbol{h}'_T - \boldsymbol{h}_t\|^2 \quad (1)$$

After updating the parameter, the entire student model, including the output layer, is updated by minimizing the distillation loss. Given the output of the student model as $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_t$, T-th output of the teacher model as $\mathbf{y}'_T$, and ground truth as $\mathbf{y}_{\text{true}}$, the distillation loss L_{KD} is calculated as follows:

$$L_{KD} = \frac{1}{t}\sum_{i=1}^{t}(\mathcal{H}(\mathbf{y}_{\text{true}}, \mathbf{y}_i)) + \lambda\mathcal{H}(\mathbf{y}'_T, \mathbf{y}_t). \quad (2)$$

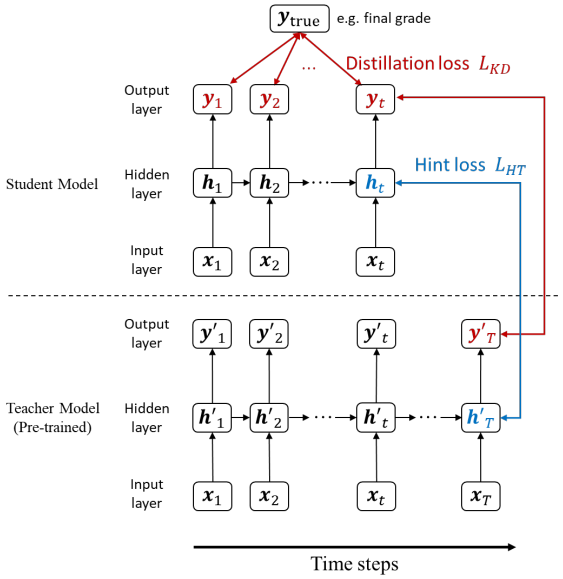


Figure 2: RNN-FitNets.

Table 1: Grade point average distribution.

GPA	A	B	C	D	F
Number of students	25	50	16	12	5

where $\mathcal{H}$ refers to the cross-entropy and λ is a hyperparameter that balances both cross-entropies.

3. EXPERIMENT

3.1 Dataset

We used the same dataset as [10]. The data were collected from the Information Science course at Kyushu University. This course started in April 2016 and 15 lectures were held weekly. Table 1 shows the grade point average of the 108 students that took this course. More than two-thirds of students received an "A" or "B." On this course, the teacher and students used a learning support system called M2B [9]. The M2B system consists of three subsystems: the learning management system, Moodle; the e-portfolio system, Mahara; and the e-book system, BookLooper. Moodle recorded students' attendance, submission of reports, and access to the course. Mahara recorded students' logbook in each lecture on the course. BookLooper recorded students' reading behavior such as turning pages, drawing highlights, and taking notes.

We also applied the feature engineering method used by [10]. The collected data were converted into active learner points, as shown in Table 2. As shown in the table, the learning behavior of each lecture was evaluated on a five-point scale (0–5). Attendance and report submission were evaluated based on whether the activities were on time, late, or not completed. The quiz was evaluated based on the ratio of collected answers. The other behaviors were evaluated by comparing the students in each lecture. Before inputting these features into the prediction model, the evaluated values were divided by 5 (i.e., they were normalized with in the range of 0 to 1).

Table 2: Criteria for active learner points.

Activities	5	4	3	2	1	0
Attendance	Attendance		Being late			Absence
Quiz	Above 80%	Above 60%	Above 40%	Above 20%	Above 10%	Otherwise
Report	Submission		Late			None
Course accesses	Upper 10%	Upper 20%	Upper 30%	Upper 40%	Upper 50%	Otherwise
Word count in Mahara	Upper 10%	Upper 20%	Upper 30%	Upper 40%	Upper 50%	Otherwise
Reading time in BookLooper	Upper 10%	Upper 20%	Upper 30%	Upper 40%	Upper 50%	Otherwise
Highlights in BookLooper	Upper 10%	Upper 20%	Upper 30%	Upper 40%	Upper 50%	Otherwise
Notes in BookLooper	Upper 10%	Upper 20%	Upper 30%	Upper 40%	Upper 50%	Otherwise
Total Actions in BookLooper	Upper 10%	Upper 20%	Upper 30%	Upper 40%	Upper 50%	Otherwise

3.2 Evaluation Criteria

We applied 5-fold cross-validation to the 108 students in the dataset. The folds were made by preserving the percentage of samples for each student's grade of "A," "B," "C," "D," and "F." After the separation, we grouped grades "A" and "B" into the "no-risk" class and grades "C," "D," and "F" into the "at-risk" class because more than two-thirds of students received "A" or "B" (see Table 1). Therefore, we conducted a binary classification between "no risk" ("A" or "B") and "at-risk" ("C," "D," or "F"). For the evaluation, we calculated the recall, precision, and F-measure values for detecting at-risk students.

3.3 Comparison Models

To investigate the effectiveness of our model, we compared the evaluation values for predicting the final grades between the following three types of models:

- **RNN baseline model**

  Training the RNN-based prediction model using the learning behavior in all lecture weeks.

- **Week-by-week model**

  Training the RNN-based prediction model using the learning behavior in each lecture week. Therefore, there were 15 independent models (trained by only first-week behavior, trained until the second week of behavior, and so on).

- **RNN-FitNets**

  Training the student model from the RNN baseline model as the teacher model in each lecture week. As with the week-by-week model, there were 15 student models.

The three types of comparison models had the same architecture. We set the batch size to one. The length of time steps took an integer from 1 to 15 when the three types of comparison models predicted students at-risk. When the models were trained, the RNN baseline model used 15 time steps and the other models used the same time steps as the prediction. The input features of the model were the active learner points shown in Table 2; therefore, the number of features was nine. For the hidden layer, we used GRU with 32 units and the activation function was tanh. The output layer had two units and the activation function was softmax. We used RMSprop optimizer [16] for the hint loss and distillation loss. In both the optimizations, we set the learning rate to 0.001. In addition, we applied L2 regularization with a parameter of 0.004 for the optimization of the weights and biases in the hidden and output layers of the RNN baseline and the week-by-week model. λ in the distillation loss (Eq. (2)) was equal to the time step; for example, when RNN-FitNets was trained using the learning behavior until the second week, we set λ to 2. All models were trained for 50 epochs.

3.4 Experimental Result

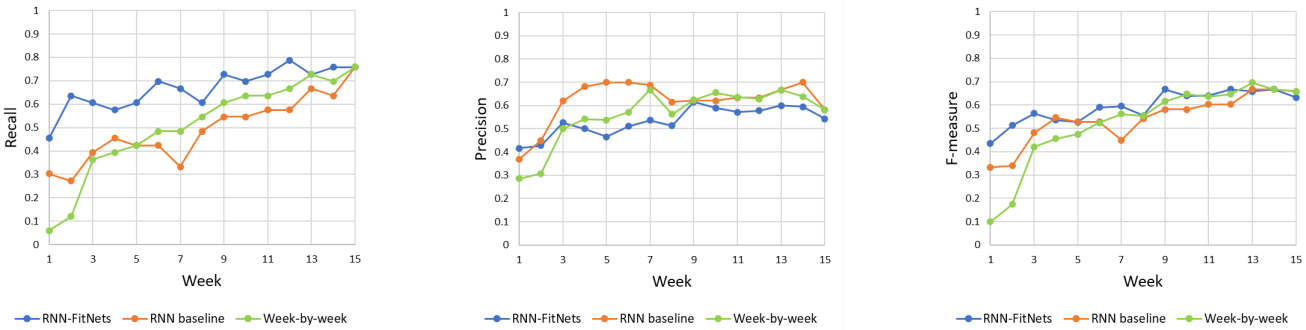
Figure 3 illustrates the evaluation of the three types of models. We summarize the results as follows:

- In most time steps, the recall values of the RNN-FitNets were higher than the values of the RNN baseline and week-by-week models. In other words, RNN-FitNets detected more at-risk students than other models.

- However, the precision values of the RNN-FitNets were lower than the value of the RNN baseline model, i.e., RNN-FitNets misdetected more no-risk students as at-risk.

- As shown by the F-measure values, the RNN-FitNets' values were higher than that of the RNN baseline and week-by-week models in most time steps. This difference was marked in early time steps. Therefore, the increase in the detection of at-risk students outweighed the increase in the misdetection, especially in early time steps.

- Comparing the evaluation values of the RNN baseline model with those of the week-by-week model, the former was superior in early time steps, although the values of the week-by-week model were close to or outperformed those of the RNN baseline model in late time steps.



(a) Recall (b) Precision (c) F-measure

Figure 3: Evaluation of three types of models.

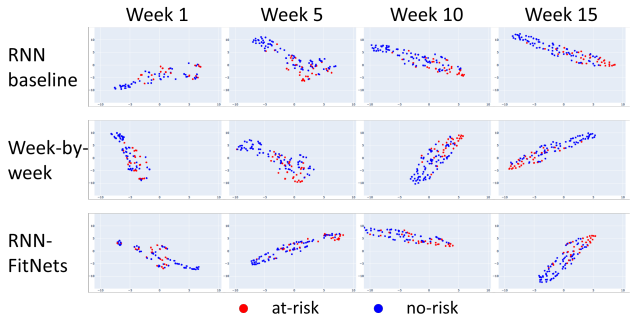


Figure 4: Visualization of the extracted feature vectors in the three types of models by t-SNE.

3.5 Discussion

The experimental result showed that the proposed models improved the detection rate of at-risk students, especially in early time steps. This improvement resulted from the distillation of time-series information. The evaluation values of the RNN baseline model were higher than those of the week-by-week model in early time steps. This result implies that the time-series information obtained by training in all the time steps is effective for early detection. In the RNN-FitNets, the time-series information was expressly passed through KD and that improved the model's performance.

To investigate whether the time-series information was distilled into the models in early time steps, we visualized the extracted feature of the three types of models. Figure 4 shows the visualization results of the feature vectors. Because the models have a 32-dimensional hidden state, we used t-SNE [17] and reduced the 32 dimensions to two dimensions for the visualization. Each point represents each feature vector for the students in the dataset. The red point is at-risk students and the blue point is no-risk students, as defined in Section 3.2. By observing the feature vectors of the RNN baseline and week-by-week models, the more time steps are used, the closer the red points are to each other and the more the shape of the mass of points becomes elongated. This means that the detection of at-risk students becomes easier in the feature vectors of late time steps. In the RNN-FitNets models, the tendency to gather red points and elongate appears in early time steps. This result shows that our KD method properly distills the time-series information extracted in the late time step.

4. CONCLUSION

In this study, we proposed RNN-FitNets, which extends FitNets, a KD method, for application to RNN architecture. RNN-FitNets transfers the time-series information extracted by the later time-step RNN into an earlier time-step RNN. Hence, the earlier time-step RNN learns the method of extracting the representative features in late time steps from short time-series data.

In the experiment, we applied RNN-FitNets to detect at-risk students in higher education. The results show that the proposed distillation model improves the detection rate of at-risk students from the base RNN models. The analysis of feature vectors indicated that our proposed model in earlier time steps extracted similar feature vectors to those of the base model in late time steps. This confirmed that our distillation strategy properly distilled the time-series information in later time steps into the model in earlier time steps.

In future work, we plan to investigate the availability of RNN-FitNets for other datasets. Moreover, we aim to formulate a new distillation method for time-series information for other models such as the Transformer model [18].

5. ACKNOWLEDGMENTS

This work was supported by JST AIP Grant Number JPMJCR19U1, and JSPS KAKENHI Grand Number JP18H04125, Japan

6. REFERENCES

[1] N. R. Aljohani, A. Fayoumi, and S.-U. Hassan. Predicting at-risk students using clickstream data in the virtual learning environment. *Sustainability*, 11(24):7238, 2019.

[2] W. Chen, C.-C. Chang, C.-Y. Lu, and C.-R. Lee. Knowledge distillation with feature maps for image classification. In *ACCV*, 2018.

[3] Y. He, R. Chen, X. Li, C. Hao, S. Liu, G. Zhang, and B. Jiang. Online at-risk student identification using rnn-gru joint neural networks. *Information*, 11(10):474, 2020.

[4] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[5] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[6] B.-H. Kim, E. Vizitei, and V. Ganapathi. Gritnet 2: Real-time student performance prediction with domain adaptation. *arXiv preprint arXiv:1809.06686*, pages 1–8, 2018.

[7] B.-H. Kim, E. Vizitei, and V. Ganapathi. Gritnet: Student performance prediction with deep learning. *arXiv preprint arXiv:1804.07405*, 2018.

[8] Q. Li, S. Jin, and J. Yan. Mimicking very efficient network for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7341–7349, 2017.

[9] H. Ogata, Y. Taniguchi, D. Suehiro, A. Shimada, M. Oi, F. Okubo, M. Yamada, and K. Kojima. M2b system: A digital learning platform for traditional classrooms in university. *Practitioner Track Proceedings*, pages 155–162, 2017.

[10] F. Okubo, A. Shimada, T. Yamashita, and H. Ogata. A neural network approach for students' performance prediction. In *LAK 2017 Conference Proceedings - 7th International Learning Analytics and Knowledge Conference*, ACM International Conference Proceeding Series, pages 598–599. Association for Computing Machinery, Mar. 2017. 7th International Conference on Learning Analytics and Knowledge, LAK 2017 ; Conference date: 13-03-2017 Through 17-03-2017.

[11] F. Okubo, T. Yamashita, A. Shimada, Y. Taniguchi, and K. Shin'ichi. On the prediction of students' quiz score by recurrent neural network. *CEUR Workshop Proceedings*, 2163, Jan. 2018. 2nd Multimodal Learning Analytics Across (Physical and Digital) Spaces, CrossMMLA 2018 ; Conference date: 06-03-2018.

[12] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[13] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[14] D. Sun, Y. Mao, J. Du, P. Xu, Q. Zheng, and H. Sun. Deep learning for dropout prediction in moocs. In *2019 Eighth International Conference on Educational Innovation through Technology (EITT)*, pages 87–90, 2019.

[15] R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, and J. Lin. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*, 2019.

[16] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[17] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[19] F. Xiong, K. Zou, Z. Liu, and H. Wang. Predicting learning status in moocs using lstm. In *Proceedings of the ACM Turing Celebration Conference-China*, pages 1–5, 2019.

Mining sequential patterns with high usage variation

Yingbin Zhang
University of Illinois at Urbana-Champaign
yingbin2@illinois.edu

Luc Paquette
University of Illinois at Urbana-Champaign
lpaq@illinois.edu

ABSTRACT
Sequential pattern mining is a useful tool in understanding learning processes, but identifying the most relevant patterns can be a challenge. Typical sequential pattern mining algorithms and interestingness metrics mainly focus on finding behavior patterns common across all students. However, educational researchers also care about individual differences. This study proposes a method for finding sequential patterns which usage have high variation across students. This method borrows techniques from the field of lag sequential analyses and meta-analyses. It uses the log odd ratio to model the individuals' usage of a sequential pattern and the heterogeneity test to examine the usage variation. We applied this method to analyzing student action logs in a virtual experimental environment and present preliminary results illustrating how the identification of sequential patterns with high usage variation provides interesting information about students' learning behavior. The proposed approach adds a way for understanding individual differences in learning processes.

Keywords
Sequential pattern mining, learning behavior differences, log odds ratio, lag sequential analysis, heterogeneity test

1. INTRODUCTION
Sequential pattern mining (SPM) aims to find the temporal associations between events [1]. For example, whether students read relevant material after answering a question incorrectly. Such sequential behaviors are named sequential patterns. SPM has shown its potentials in helping researchers understand learning behavior [2, 3].

However, there are challenges when applying SPM in education. One important challenge is that SPM algorithms may generate excessive sequential patterns, most of which are uninteresting or irrelevant to the research purpose [2]. This increases the difficulty of making meaningful interpretations and producing actionable pedagogical insights. To address this challenge, researchers select sequential patterns using interestingness metrics, such as the support value e.g., [4, 5]. The support value of a sequential pattern is the proportion of students that shown this pattern. As such, patterns with high support values will reflect similarities in the learners' behavior.

Yingbin Zhang and Luc Paquette "Mining sequential patterns with high usage variation". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 704-707. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Educational researchers also care about differences among students [6]. The understanding of individual differences in learning is essential for providing learners with adaptive scaffolding. To address this need, this study proposes a method borrowing from lag sequential analyses and meta-analyses that uses log odd ratio and the heterogeneity test to select sequential patterns based on their variation in usage across learners.

2. Methodology
Let $E = \{e_1, e_2, \dots, e_p\}$ be a set of p unique events that may occur within a specific learning environment, such as answering a question and asking for a hint. Let $S_m = \{i_1, i_2, \dots, i_n\}$ be a sequence of N temporally ordered items with each i_j being a subset of E. A sequence is a student's learning process data, such as action logs in an intelligent tutoring system. Each i_j usually contains one event because students rarely initiate two different actions simultaneously. Let $e_x \rightarrow e_y$ be a sequential pattern where e_y occurs after e_x (e_x and e_y may be the same event). Let $\overline{e_x}$ denote an event other than e_x. If there are $i_k = e_x, i_l = e_y$, and $k < l$ in S_m, S_m contains $e_x \rightarrow e_y$ [8].

2.1 Using log odds ratio to model sequential pattern usage
If we fix the gap between $e_x \rightarrow e_y$ to a constant c, we may use methods from the field of lag-sequential analyses to quantify students' usage on $e_x \rightarrow e_y$ [8]. Fixing the gap to c means that we only consider $i_k = e_x, i_l = e_y$, and $l - k = c$ as an occurrence of $e_x \rightarrow e_y$. For example, $c = 1$ means that we only count the case where e_y directly follows e_x. Lag-sequential analysis utilizes statistics from contingency table analyses to quantify the usage of $e_x \rightarrow e_y$, such as the odds ratio and the log odds ratio [8].

Let the frequency of pairs of consecutive events where the first event is e_x and the second event is e_y $n(e_x \rightarrow e_y) = a_m$, Let the frequency of pairs of consecutive events where the first event is e_x but the second event is not e_y $n(e_x \rightarrow \overline{e_y}) = b_m$. Let the frequency of pairs of consecutive events where the first event is not e_x but the second event is e_y $n(\overline{e_x} \rightarrow e_y) = c_m$. Let the frequency of pairs of consecutive events where the first event is not e_x and the second event is not e_y $n(\overline{e_x} \rightarrow \overline{e_y}) = d_m$. The odds ratio of $e_x \rightarrow e_y$ in S_m can be calculated as $\frac{a_m d_m}{b_m c_m}$, while the log odds ratio is $log\frac{a_m d_m}{b_m c_m}$. However, there is measurable bias in this expression when the sample is small. A slightly modified version is often used to reduce bias [9]:

$$Y_m(e_x \rightarrow e_y) = log\frac{(a_m + \frac{1}{2})(d_m + \frac{1}{2})}{(b_m + \frac{1}{2})(c_m + \frac{1}{2})}. \quad (2)$$

The log odds ratio of $e_x \rightarrow e_y$ represents the relative likelihood that e_y occurs after e_x during a student's learning, considering the

probability that e_y occurs after an event other than e_x. If a sequential pattern contains more than two events, researchers may segment a sequential pattern into two sub-patterns and represent the sequential pattern as one sub-patterns follows another. For example, $e_x \rightarrow e_y \rightarrow e_z \rightarrow e_w$ may be represented as $e_{x \rightarrow y} \rightarrow e_{z \rightarrow w}$. This preprocessing has been used in computing the confidence value of sequential patterns longer than two events [10]. Then, the above procedure can be used to calculate the log odds ratio.

The variance of $Y_m(e_x \rightarrow e_y)$ is:

$$V_m(e_x \rightarrow e_y) = \frac{1}{a_m + \frac{1}{2}} + \frac{1}{b_m + \frac{1}{2}} + \frac{1}{c_m + \frac{1}{2}} + \frac{1}{d_m + \frac{1}{2}}. \quad (3)$$

$V_m(e_x \rightarrow e_y)$ characterizes the imprecision of the log odds ratio and decreases as the length of S_m increases. The log odds ratio based on a long sequence is more precise than that based on a short sequence [9].

2.2 Ranking sequential patterns by variation across users

We can examine whether the log odds ratio varies across participants via the heterogeneity test used in meta-analyses [11]. One commonly used heterogeneity test is the Q test [12]. In meta-analyses, Q is the weighted sum of the squared deviations of each study's effect estimate from the weighted mean of all studies' effect estimates. The weighting for each study is the inverse of the variance of the study's effect estimate. Thus, in terms of the variation of the log odds ratio of $e_x \rightarrow e_y$, Q can be calculated using the formula:

$$Q(e_x \rightarrow e_y) = \sum \frac{(Y_m - \overline{Y})^2}{V_m}, \quad (4)$$

where $\overline{Y}$ is the weighted mean of log odds ratios, i.e.,

$$\overline{Y} = \frac{\sum \frac{Y_m}{V_m}}{\sum \frac{1}{V_m}}. \quad (5)$$

Q follows a chi-square distribution with $k - 1$ degrees of freedom, where k is the number of sequences or participants. Thus, if $Q(e_x \rightarrow e_y)$ is higher than the critical value for a given significance level (e.g., 0.05), we may conclude that the usage of $e_x \rightarrow e_y$ has statistically significant variation across participants. Moreover, for the same dataset, the number of participants is constant, and thus, the Qs of all sequential patterns follow the same chi-square distribution and are comparable. However, it is difficult to interpret Q because its magnitude is influenced by the number of participants. The I^2 index overcomes this issue [13].

$$I^2 = \begin{cases} \frac{Q - (k-1)}{Q} * 100\%, if\ Q > (k-1) \\ 0, else \end{cases}. \quad (6)$$

$I^2(e_x \rightarrow e_y)$ can be interpreted as the proportion of variation in the log odds ratio of $e_x \rightarrow e_y$ due to true between-participants variance. Ranking sequential patterns by Q and I^2 produces the same results because k is fixed for the same dataset.

3. Example

This section applied the proposed method to a dataset of student action logs collected from a virtual experiment environment called LabBuddy [14].

3.1 Data

3.1.1 Participants

The data were collected from a graduate-level enzymology course at a university in the Netherlands. Participants were 76 graduate students in this course. The average age was 22.91 years old (SD = 1.80). Around 64.47% of the students were female.

3.1.2 LabBuddy

The course helped students prepare for the laboratory classes using LabBuddy. LabBuddy in this course contained a self-directed learning task, which included six research questions offered by a virtual tutor, Professor Kabel. Students start with proposing hypotheses for each question and make an experimental design via a flow chart to test the hypotheses (Figure 1). Each block in the flow chart represents a chemical method and contains details about the method. Each block also contains some closed questions that students must answer correctly before implementing the method and getting the raw data. Students do some calculations based on the raw data to get the results. The details, raw data, and calculations of a method are located in different subblocks of a block. If students are struggling with a closed question, they may request hints or the correct answer. Once students obtain the results, they may consult Professor Kabel to interpret them and either accept or reject their hypotheses. Students used LabBuddy for an average of 7.5 h distributed over three days. Their action logs were used for analysis.

3.2 Analyses

We preprocessed the action logs by removing redundant successive repeated actions (e.g., multiple selections of the same block) and contextualizing some actions (e.g., is the submitted answer to a closed question correct?). The preprocessing resulted in 19 unique events. The average number of events in a student's action log was 995 (SD = 363). Then, we implement our methods via the following procedure:

1. Apply the cSPADE algorithm to find frequent sequential patterns with support no less than 0.5. We used this algorithm because it allows us to fix the gap between events in a sequential pattern, a prerequisite for calculating the log odds ratios. The gap was fixed to 1 in the analysis. For simplicity, we only focused on sequential patterns containing two events. This step generated 81 frequent sequential patterns.
2. For each student, compute the log odds ratio, variance, and the number of occurrences of each frequent sequential pattern.
3. For each frequent sequential pattern, conduct the Q test and calculate the I^2 index, the average log odds ratio, and the average occurrence. As the Q test was run 81 times, we used the Benjamini-Yekutieli correction to control the false discovery rate [15].

Note we only apply our method to frequent sequential patterns because the variation of a sequential pattern across participants would be low if few participants used a pattern (i.e., it was infrequent).

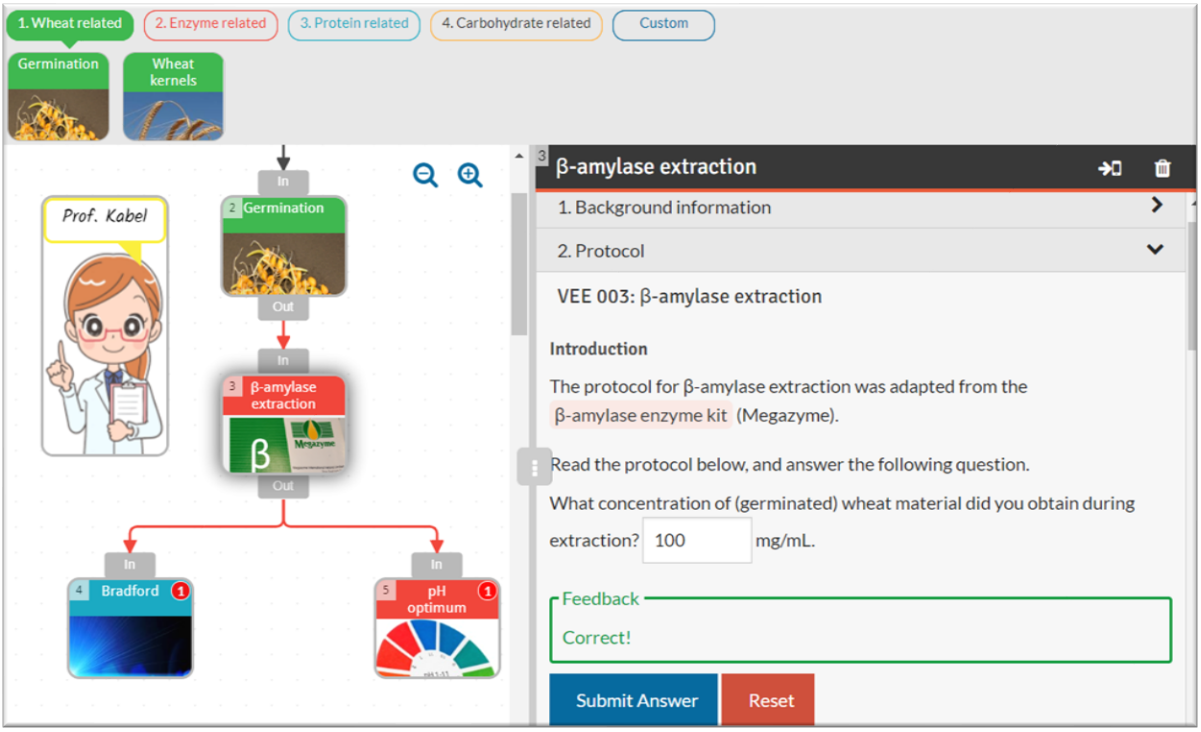


Figure 1. The LabBuddy learning environment.

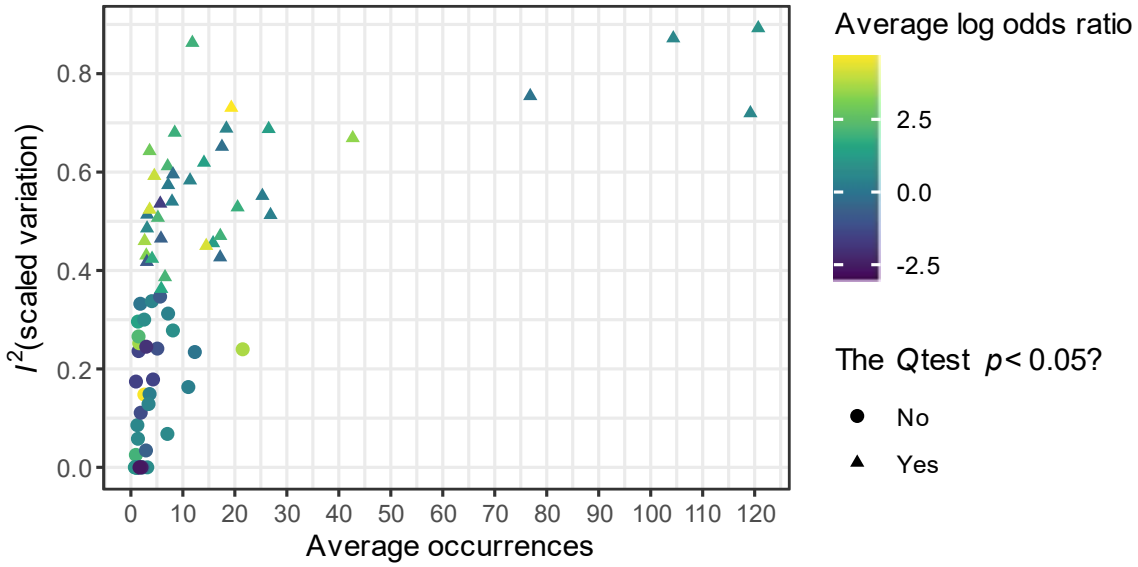


Figure 2. The I^2 indexes, average log odds ratios, and average occurrences of sequential patterns.

3.3 Preliminary results

Figure 2 visualizes the relationships among the I^2 indexes, average log odds ratios, and average occurrences of the 81 sequential patterns. There were moderate positive relationships between the I^2 index and the average log odds ratio ($r = 0.57$, $p < 0.001$) as well as the average occurrences ($r = 0.36$, $p = 0.001$). Nevertheless, ranking sequential patterns by their variation between students results in a different set of selected patterns than ranking them by their similarities (average log odds ratios and occurrences) between students. Some sequential patterns had few average occurrences (e.g., less than 5) or negative average log odds ratios while still being used differentially by students (the adjusted p of the Q test < 0.05). Some sequential patterns had relatively high average occurrences (e.g., larger than 10) or average log odds ratios (e.g., larger than 1) but were used consistently across students (the adjusted p of the Q test > 0.05).

We investigated how I^2 might help us detect behavioral differences by looking more closely at two sequential patterns with distinct I^2: *Submitting an intermedia answer → Submitting an intermedia answer* and *Requesting a hint → Requesting a hint*. Both patterns had high values in average occurrences and log odds ratios (see Table 1). *Submitting an intermedia answer → Submitting an*

intermedia answer had a I^2 of 0.75 ($p < 0.001$), indicating that students had high variations in the usage on this pattern. Further analysis showed that, in 9.54% of all pairs of students (309/3,240), the log odd ratio was significantly different between the two students. This means that among 10 randomly sampled pairs, on average, there was one pair where the two students had significantly different probability of submitting two intermedia answers consecutively. In contrast, the usage on *Requesting a hint → Requesting a hint* was relatively consistent across students (I^2 = 0.24, p = 0.52). Analyses showed that, in only 1.6% of pairs, the log odd ratio was significantly different between two students.

Table 1. The metrics of two sequential patterns

Pattern	I^2	p for the Q test	Log odds ratio	Occurrences
RH.RH	0.24	0.52	3.76	21.51
SI.SI	0.75	0.00	4.81	19.32

Note. RH.RH: *Requesting a hint → Requesting a hint*. SS.WA: *Submitting an intermedia answer → Submitting an intermedia answer*.

4. Discussion

This study proposed a method for mining sequential patterns which usage has high variation across students. We applied the method to a dataset of student action logs in a virtual experimental environment. The preliminary results suggest that ranking sequential patterns by their variation across students results in a different selection of patterns than by their similarities across students. Moreover, the results demonstrated how the proposed method could capture individual differences in sequential behavior patterns. The approach adds a way for understanding individual differences in learning, which is critical in education.

The next step is to examine whether the sequential patterns with high variation are related to students' learning gains. Such investigation would contribute to our understanding of how differences in which sequential patterns may lead to differences in learning outcomes. The insights, in turn, would provide information about how the learning environment might scaffold the learners' interaction with the learning environment by prompting sequential behavior patterns beneficial to learning and discouraging patterns harmful to learning.

Our approach requires fixing the gap between events of a sequential patterns. This requirement limits flexibility. For example, researchers may regard *Submitting an intermedia answer → Requesting a hint → Submitting an intermedia answer* as an instance of *Submitting an intermedia answer → Submitting an intermedia answer*, but fixing the gap to 1 excludes this possibility. On the other hand, if fixing the gap to 2, *Submitting an intermedia answer* directly after *Submitting an intermedia answer* would not be regarded as an instance of *Submitting an intermedia answer → Submitting an intermedia answer*. The limitation is the same as the issue that the lag between the antecedent and consequent events must be fixed in a lag sequential analysis [8]. Addressing this issue is challenging but worthy of effort.

5. ACKNOWLEDGMENTS

We would like to thank the Laboratory of Food Chemistry, Wageningen University & Research for allowing us to use the dataset and access to LabBuddy.

6. REFERENCES

[1] Baker, R. 2010. Data mining for education. In B. McGaw, P. Peterson & E. Baker (Eds), *International Encyclopedia of Education* (pp. 112-118). Oxford, UK: Elsevier Ltd.

[2] Zhou, M., Xu, Y., Nesbit, J. C. & Winne, P. H. 2010. Sequential pattern analysis of learning logs: Methodology and applications. In *Handbook of Educational Data Mining* (pp. 107-121). Boca Raton: CRC Press.

[3] Moon, J. & Liu, Z. 2019. Rich representations for analyzing learning trajectories: Systematic review on sequential data analytics in game-based learning research. In A. Tlili & M. Chang (Eds), *Data analytics approaches in educational games and gamification systems* (pp. 27-53). Singapore: Springer.

[4] Jiang, Y., Paquette, L., Baker, R. S. & Clarke-Midura, J. 2015. Comparing novice and experienced students within virtual performance assessments. In *Proceedings of the 8th International Conference on Educational Data Mining* (Madrid, Spain, Jun. 2015). International Educational Data Mining Society, 136-143.

[5] Kang, J., Liu, M. & Qu, W. 2017. Using gameplay data to examine learning behavior patterns in a serious game. *Comput. Hum. Behav.*, 72, (2017), 757-770. http://doi.org/10.1016/j.chb.2016.09.062

[6] Malmberg, L., Lim, W. H., Tolvanen, A. & Nurmi, J. 2016. Within-students variability in learning experiences, and teachers' perceptions of students' task-focus. *Frontline Learning Research*, 4, 5 (2016), 62-82. http://doi.org/10.14786/flr.v4i5.227

[7] Agrawal, R. & Srikant, R. 1995. Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering* (Taipei, Taiwan, 1995). IEEE, 3-14.

[8] Bakeman, R. & Quera, V. *Sequential analysis and observational methods for the behavioral sciences.* Cambridge University Press, New York, NY, US, 2011.

[9] Dagne, G. A., Howe, G. W., Brown, C. H. & Muthén, B. O. 2002. Hierarchical modeling of sequential behavioral data: An empirical Bayesian approach. *Psychol. Methods*, 7, 2 (Jun. 2002), 262-280. http://doi.org/10.1037/1082-989X.7.2.262

[10] Fournier-Viger, P., Faghihi, U., Nkambou, R. & Nguifo, E. M. 2012. CMRules: Mining sequential rules common to several sequences. *Know.-Based Syst.*, 25, 1 (2012), 63-76. http://doi.org/10.1016/j.knosys.2011.07.005

[11] Huedo-Medina, T. B., Sánchez-Meca, J., Marín-Martínez, F. & Botella, J. 2006. Assessing heterogeneity in meta-analysis: Q statistic or I² index? *Psychol. Methods*, 11, 2 (2006), 193-206. http://doi.org/10.1037/1082-989X.11.2.193

[12] Cochran, W. G. 1954. The Combination of Estimates from Different Experiments. *Biometrics*, 10, 1 (1954), 101-129. http://doi.org/10.2307/3001666

[13] Higgins, J. P. T. & Thompson, S. G. 2002. Quantifying heterogeneity in a meta-analysis. *Stat. Med.*, 21, (2002), 1539-1558. http://doi.org/10.1002/sim.1186

[14] Van Der Kolk, K., Beldman, G., Hartog, R. & Gruppen, H. 2012. Students Using a Novel Web-Based Laboratory Class Support System: A Case Study in Food Chemistry Education. *J. Chem. Educ.*, 89, 1 (Jan. 2012), 103-108. http://doi.org/10.1021/ed1005294

[15] Benjamini, Y. & Yekutieli, D. 2001. The Control of the False Discovery Rate in Multiple Testing under Dependency. *The Annals of Statistics*, 29, 4 (Jan. 2001), 1165-1188.

Demonstrating REACT: a Real-time Educational AI-powered Classroom Tool*

Ajay Kulkarni
George Mason University
akulkar8@gmu.edu

Olga Gkountouna
George Mason University
ogkounto@gmu.edu

ABSTRACT

We present a demonstration of REACT, a new **R**eal-time **E**ducational **A**I-powered **C**lassroom **T**ool that employs EDM techniques for supporting the decision-making process of educators. REACT is a data-driven tool with a user-friendly graphical interface. It analyzes students' performance data and provides context-based alerts as well as recommendations to educators for course planning. Furthermore, it incorporates model-agnostic explanations for bringing explainability and interpretability in the process of decision making. This paper demonstrates a use case scenario of our proposed tool using a real-world data set, and presents the design of its architecture and user-interface. This demonstration focuses on the agglomerative clustering of students based on their performance (i.e., incorrect responses and hints used) during an in-class activity. This formation of clusters of students with similar strengths and weaknesses may help educators to improve their course planning by identifying at-risk students, forming study groups, or encouraging tutoring between students of different strengths.

Keywords

Clustering, Decision-support, Educational tool, Explainability, Human-centered computing

1. INTRODUCTION

Instructors play a crucial role in educational institutions, where one of their main responsibilities is effective high-quality teaching. To do so they must stay updated with students' responses, efforts, and outcomes, in order to provide timely feedback to promote students' improvement [9, 29]. One of the ways this can be achieved is by clustering students into groups based on various characteristics such as their learning style preferences, academic performance, behavioral interaction, etc., which can be utilized to explore collaborative learning opportunities and identify at-risk students at an early stage [3]. Thus, this creates a need for tools that will empower instructors to achieve these objectives in the classroom. To this end, the fields of Educational Data Mining (EDM) and Learning Analytics (LA) have emerged with the goal to understand how educational data can benefit the science of learning [7]. One of the ways to promote this understanding is to use AI-powered real-time visualizations. These visual displays summarize large amounts of data in a meaningful way. This is important for humans' sense-making and decision-making [35] as it helps human cognition [12]. An example of these visual displays are dashboards which may contain various data indicators [20].

Furthermore, applications of Artificial Intelligence (AI) in the domain of education for predicting student performance, detecting undesirable student behavior, or providing feedback for supporting instructors and students, are becoming more common [8]. This creates a need for incorporating interpretability, explainability, and, ultimately, trustworthiness in AI for supporting human teaching and learning [39]. The simplest way to include explainability in AI is by using model-agnostic explanations that consist of textual and visual explanations [4]. Interpretability can be achieved by including humans in the process of decision making (HitAI) i.e., decision power is given to the specialized professionals who utilize machines/tools as advisors [41].

This paper presents a demonstration of REACT, a **R**eal-time **E**ducational **A**I-powered **C**lassroom **T**ool, which utilizes the principles of HitAI and model-agnostic explanations to support educators in their decision-making. REACT clusters students based on their responses during in-class activities, and provides context-based recommendations for course planning. It also provides personalized feedback about individual students. REACT is a real-time data-driven decision-support tool that incorporates explainability, interpretability, and portability. It presents different indicators of students, their learning processes, learning contexts, and recommendations for increasing efficiency in course management. Based on the Learning Analytics Process model [38], REACT may directly help educators in awareness, reflection, and sense-making while it can indirectly create impact and motivate to take actions. This functionality can support educators in making decisions concerning their course planning and instructional goals setting, by consistently monitoring students' activities (tests, quizzes, and exercises) to inspect the their learning process.

*(Does NOT produce the permission block, copyright information nor page numbering). For use with ACM_PROC_ARTICLE-SP.CLS. Supported by ACM.

Ajay Kulkarni and Olga Gkountouna "Demonstrating REACT: a Real-time Educational AI-powered Classroom Tool". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 708-712. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

The remainder of this paper is structured as follows. Section 2 presents the background of EDM and how clustering can be useful in this context. Section 3 describes the architecture of our tool and explains how clustering is utilized in REACT. Section 4 presents the details on the design of the user interface and details of the demonstration. Finally, Section 5 concludes the paper discussing directions of future research.

2. BACKGROUND

EDM enhances the decision-making of teachers, students, and educational institutes by utilizing data mining techniques in an educational context [31]. A variety of methods, including cluster analysis, outlier detection, text mining, recommendation systems, and visualizations can be applied in the EDM domain [32]. *Cluster analysis* or *clustering* is the most well known unsupervised machine learning task [24]. In the context of EDM, identifying meaningful clusters of students can be useful in understanding their learning behavior [14, 13, 10, 5, 26, 22]. Clustering consists of four steps [40]: (1.) *Feature extraction and selection* – Relevant features are selected from the data and transformed into an appropriate format. (2.) *Algorithm design* – A suitable clustering algorithm and (dis)similarity measure are selected. (3.) *Evaluation* – Different clustering results can be evaluated using different metrics such as external, internal and/or relative indices. (4.) *Explanation* - The main purpose of clustering is to generate knowledge, useful for decision-making. This is conveyed to the user in different ways such as visualizations, textual feedback, or statistical metrics.

Hierarchical clustering organizes the data points in a taxonomy tree of clusters and sub-clusters [37]. Thus, it is suitable for detecting clusters of arbitrary shape, type and hierarchical relationships [40]. Hierarchical clustering has been shown to provide good results for small datasets [1], which is useful for typical class sizes. Furthermore, the entire clustering process can be visualized by plotting a *dendrogram*, which shows the cluster-subcluster relationships, similarity between clusters, and the order in which they are merged [37]. This results in an informative visualization of the data clustering structures [40], fulfilling the goal of explainability. There are two basic approaches to Hierarchical clustering – agglomerative and divisive [25]. The agglomerative hierarchical clustering is a bottom-up approach that starts with each points being an individual cluster, and merges the closest pair of clusters at each step. The divisive method is top-down approach that starts with points being in one large cluster and progressively divides them. This is computationally expensive [15] and not commonly used [40]. Thus, agglomerative hierarchical clustering makes a suitable choice for implementing cluster analysis on REACT.

3. REACT ARCHITECTURE

REACT is developed using the `R Shiny` framework which incorporates the principles of reactive programming [6] that are suitable for interactive applications. REACT is portable in a sense that it can be connected to any Learning Management System (LMS), like Moodle or Blackboard, as well as different database management systems, including MySQL, Oracle, Salesforce, etc. This can be achieved by using different packages such as DBI[1] (for databases), bRush[2] and rcanvas[3] (for the Canvas LMS) which are available in R. Additionally, many other LMSs offer REST APIs which can be connected with REACT using httr[4] and jsonlite[5] packages.

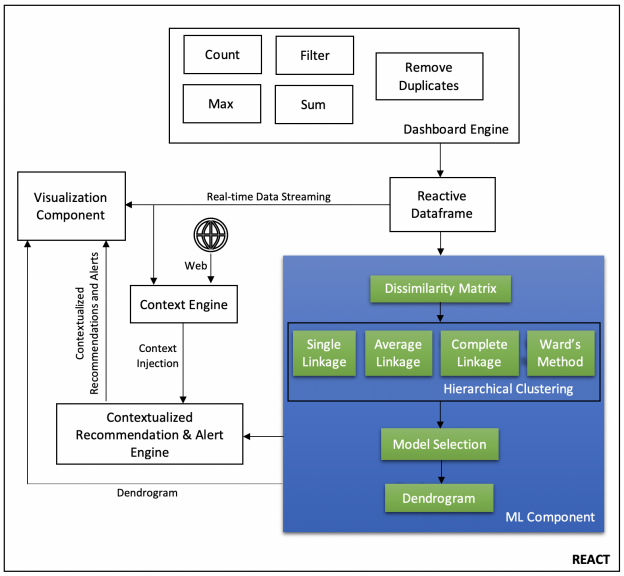


Figure 1: Architecture of REACT

The architecture of REACT is motivated from RAED [23] and it is shown in Figure 1. It consists of five main components: the Dashboard Engine, the Machine Learning (ML) Component, the Context Engine, the Contextualized Recommendation & Alert Engine, and the Visualization Component. Due to space limitations, we focus on the component that implements clustering as an EDM technique, i.e., the ML component.

The ML component receives input from a reactive data frame that contains the input features of each student and initiates the clustering process by first calculating all the pairwise distances (i.e., dissimilarities) of students. We use the Gower distance [18] as the dissimilarity metric for the clustering, as it can be applied to mixed data (i.e., a mix of numerical and categorical variables) in general [33]. However, for the purposes of this demonstration, we use as input features of each student the numbers of incorrect responses and the number of hints used per learning concept. For these numerical features, Gower uses Manhattan distance to calculate dissimilarity. The Dissimilarity Matrix sub-component calculates the pairwise distances between all n observations (i.e., students) in the data set organized in an $n \times n$ matrix, using the `daisy() R` function. This dissimilarity matrix then becomes the input of the Hierarchical Clustering sub-component.

The Hierarchical Clustering sub-component trains four different hierarchical clustering models using the same dissim-

[1] `https://dbi.r-dbi.org`
[2] `https://github.com/erikpal/bRush`
[3] `https://github.com/daranzolin/rcanvas`
[4] `https://github.com/r-lib/httr`
[5] `https://github.com/jeroen/jsonlite`

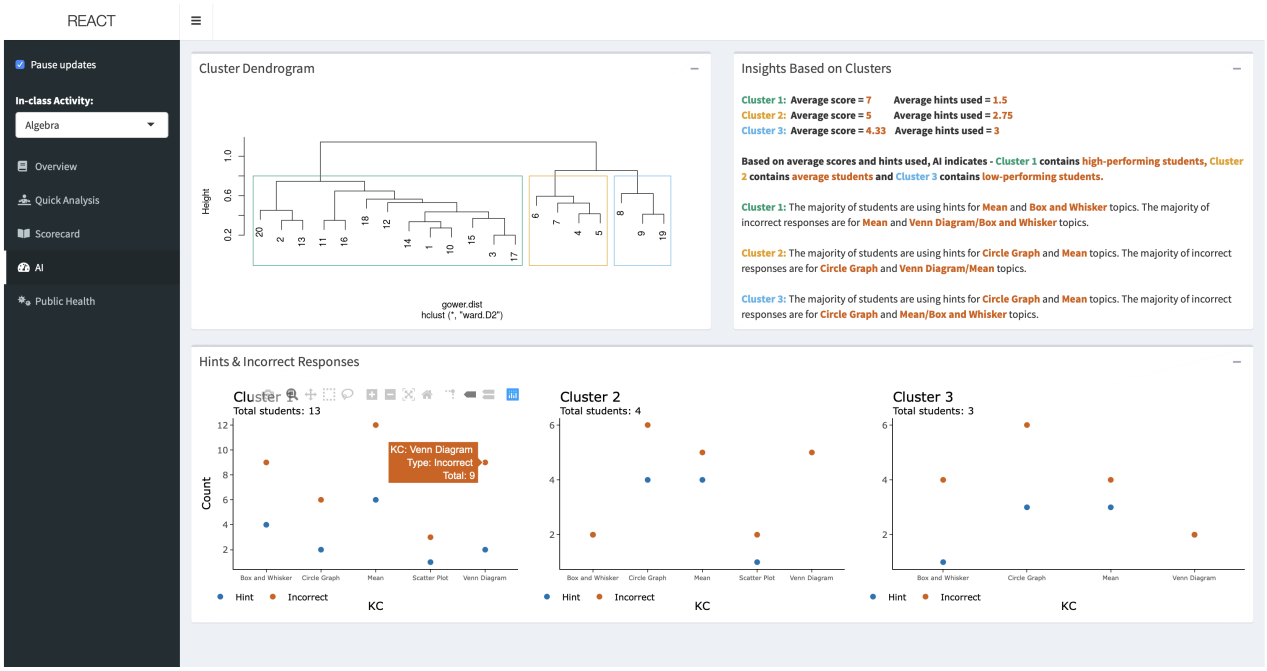


Figure 2: The AI tab provides real-time insights of clustering to instructors with visual explanation (dendrogram - top left) and textual template-based recommendations (top right)

ilarity matrix. These models are based on four different linkage methods: Single linkage, Average linkage, Complete linkage, and Ward's method. The `R` function `agnes()` is used for building these models and computing their agglomerative coefficients.

The Model Selection sub-component ensures robustness and acts as an internal index for evaluations. It compares the four clustering results based on their agglomerative coefficients. Their values lie between 0 to 1, and describe the strength of the corresponding clustering structure [21]. This sub-component selects the model with the highest agglomerative coefficient.

The Dendrogram sub-component creates a visualization of the hierarchy of clusters and sub-clusters that are the result of the selected model. This visualized hierarchy is called a *dendrogram*. The dendrogram provides a diagrammatic representation of the hierarchical cluster analysis. It can help to understand the clustering process which may help to incorporate explainability. An example of a dendrogram is shown in Figure 2 and discussed in the next section.

4. DESIGN AND DEMO

A dashboard can be defined as "an easy to read, often single page, real-time user interface, showing a graphical presentation of the current status (snapshot) and historical trends of an organization's key performance indicators (KPIs) to enable instantaneous and informed decisions to be made at a glance" [11]. It is also common for decision-makers to use KPIs for understanding the performance or the deviation from the set target at a glance [28]. Thus, the user-interface of REACT is designed as an interactive dashboard, displaying KPIs to help teachers monitor and understand their student's learning performance. These KPIs include the minimum, maximum, median and mean scores of the class, as well as the number of students who have completed all the questions of the in-class activity thus far.

4.1 Design Elements

A dashboard's visual attraction significantly affects its perceived usefulness and its potential to bring change in users' behavior [27]. The design choices of REACT were made with this consideration in mind. The selection of visualizations is based on the chart suggestions provided by Abela [2] and a review provided by Schwendimann et al. [34]. Further, its color palettes were selected so that it is colorblind-friendly [17]. REACT contains interactive visualizations and tables. Interactive applications need to ensure that they are easy to learn, and effective as well as enjoyable to use [30]. To ensure this, we aimed to follow the 'golden rules' of interface design proposed by Shneiderman et al. [36] – strive for consistency, permit easy reversal of actions, keep users in control, and reduce their short-term memory load.

The user interface of REACT currently has five tabs:
Overview – presents the KPIs, an interactive plot for monitoring students' performance, and textual alerts & recommendations for the instructors.

Quick Analysis – presents an interactive plot that monitors the progress of students in real time, and bar charts that count the incorrect responses, and hints used for each KC.

Scorecard – displays a histogram of the score distribution, and a dynamic table with students' information and scores, both updated in real-time.

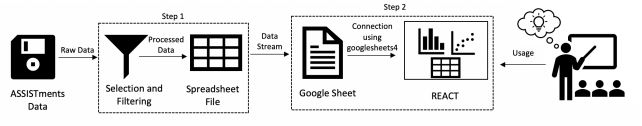


Figure 3: Creating a real-time demo of REACT

AI – provides insights of the cluster analysis and textual template-based recommendations. Figure 2 shows a screenshot of this tab. An instructor may use this tab to see on the dendrogram (top left) how different groups (i.e., clusters) of students are formed, based on their performance in in-class activities. Textual explanations about each of these groups are provided on the top right. Finally, to enhance interpretability, each of the different clusters is also visually explored at the bottom of the tab. The counts of incorrect answers and the counts of hints used are displayed per Knowledge Component (KC) for each group of students.

Public Health – provides context based on the COVID-19 outbreak. It displays infection rates in the surrounding counties and counts of students who may live in high risk areas, to inform educators, who may opt to transition online.

4.2 Demo

Holstein et al. [19] note the importance of using real-world datasets to understand the behavior of LA tools. We use the 2009-2010 Skill-builder ASSISTments data set [16]. The raw data consists of more than 100,000 rows representing details of 4217 students and 111 Knowledge Components (KCs). To achieve the objective of this demonstration, we randomly selected a sample of 20 students. Due to privacy, this data set includes only pseudo-ids. In real-word uses of REACT, authenticated instructors will be able to see students' names, as memorising their ids would be troublesome. Our approach to create a demonstration of REACT is shown in Figure 3 and can be summarized in the following steps:

- **Step 1 (Filter):** We selected 20 students and two questions from five KCs from the topic of statistics (Mean, Circle Graph, Venn Diagram, Box and Whisker Plot, and Scatter Plot). This filtered data set is first stored in a spreadsheet on a local hard disk.
- **Step 2 (Stream):** The filtered data from *Step 1* are then streamed on a Google sheet that acts as a database for this demonstration. It is connected to REACT using the `googlesheets4`[6] package.
- **Step 3 (Use):** REACT receives live updates from the streaming data. These concern hints that each student uses during the simulated in-class activity, as well as if they provided a correct or incorrect response to each question, as time progresses. These data are processed on the fly and used to update the visualisations, alerts, and recommendations displayed on the user interface.

A live version of REACT[7] is deployed using the `Shiny Server` and it can be accessed using a web browser on any desktop, laptop, tablet, or smartphone.

[6] `https://googlesheets4.tidyverse.org`
[7] `https://tinyurl.com/y7cbbbej`

5. CONCLUSIONS AND FUTURE WORK

We presented REACT, a data-driven, visual, decision-support tool that incorporates model-agnostic explanations. This paper provides details on demonstrating a use-case scenario by utilizing the ASSISTments dataset. Our next step is to evaluate our proposed tool with the help of domain experts, using a combined approach of think-aloud testing and questionnaires. This approach will help us to understand the usability and user experience while interacting with REACT. The results from this combined approach can help us to identify directions of improvement in the interface design and to propose the addition of new features. In the future, we aim to answer *how the integration of AI and visualizations in real-time can impact the instructors' decision-making process, and to what extent they do trust it.* The answers to these questions will play a crucial role in making REACT a deployable tool that can enhance data-driven decision-making in education.

6. REFERENCES

[1] O. A. Abbas. Comparisons between data clustering algorithms. International Arab Journal of Information Technology (IAJIT), 5(3), 2008.
[2] A. Abela. Chart suggestions-a thought starter. Revisado el, 20, 2006.
[3] H. Aldowah, H. Al-Samarraie, and W. M. Fauzy. Educational data mining and learning analytics for 21st century higher education: A review and synthesis. Telematics and Informatics, 37:13–49, 2019.
[4] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Information Fusion, 58:82–115, 2020.
[5] E. Ayers, R. Nugent, and N. Dean. Skill set profile clustering based on weighted student responses. In EDM, pages 210–217, 2008.
[6] E. Bainomugisha, A. L. Carreton, T. v. Cutsem, S. Mostinckx, and W. d. Meuter. A survey on reactive programming. ACM Computing Surveys (CSUR), 45(4):1–34, 2013.
[7] R. S. Baker and P. S. Inventado. Educational data mining and learning analytics. In Learning analytics, pages 61–75. Springer, 2014.
[8] R. S. Baker and K. Yacef. The state of educational data mining in 2009: A review and future visions. Journal of Educational Data Mining (JEDM), 1(1):3–17, 2009.
[9] P. Black and D. Wiliam. Assessment and classroom learning. Assessment in Education: principles, policy & practice, 5(1):7–74, 1998.
[10] F. Bouchet, J. M. Harley, G. J. Trevors, and R. Azevedo. Clustering and profiling students according to their interactions with an intelligent tutoring system fostering self-regulated learning. Journal of Educational Data Mining (JEDM), 5(1):104–146, 2013.
[11] F. Brouns, M. E. Zorrilla Pantaleón, E. E. Álvarez Saiz, P. Solana-González, Á. Cobo Ortega, E. R. Rocha Blanco, M. Collantes Viaña, C. Rodríguez Hoyos, M. De Lima Silva,

C. Marta-Lazo, et al. Eco d2.5 learning analytics requirements and metrics report. 2015.

[12] M. Card. Readings in information visualization: using vision to think. Morgan Kaufmann, 1999.

[13] A. Dutt, S. Aghabozrgi, M. A. B. Ismail, and H. Mahroeian. Clustering algorithms applied in educational data mining. International Journal of Information and Electronics Engineering, 5(2):112, 2015.

[14] A. Dutt, M. A. Ismail, and T. Herawan. A systematic review on educational data mining. IEEE Access, 5:15991–16005, 2017.

[15] B. S. Everitt, S. Landau, and M. Leese. Cluster analysis arnold. A member of the Hodder Headline Group, London, pages 429–438, 2001.

[16] M. Feng, N. Heffernan, and K. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. User Modeling and User-Adapted Interaction, 19(3):243–266, 2009.

[17] S. Few. Information dashboard design: The effective visual communication of data. O'Reilly Media, Inc., 2006.

[18] J. C. Gower. A general coefficient of similarity and some of its properties. Biometrics, pages 857–871, 1971.

[19] K. Holstein, B. M. McLaren, and V. Aleven. Co-designing a real-time classroom orchestration tool to support teacher–ai complementarity. Journal of Learning Analytics, 6(2):27–52, 2019.

[20] M. Ji, C. Michel, E. Lavoué, and S. George. Ddart, a dynamic dashboard for collection, analysis and visualization of activity and reporting traces. In European Conference on Technology Enhanced Learning, pages 440–445. Springer, 2014.

[21] L. Kaufman and P. J. Rousseeuw. Partitioning around medoids (program pam). Finding groups in data: an introduction to cluster analysis, 344:68–125, 1990.

[22] N. A. Khayi and V. Rus. Clustering students based on their prior knowledge. International Educational Data Mining Society, 2019.

[23] A. Kulkarni and M. Eagle. Towards understanding the impact of real-time ai-powered educational dashboards (raed) on providing guidance to instructors.

[24] T. S. Madhulatha. An overview on clustering methods. arXiv preprint arXiv:1205.1117, 2012.

[25] A. Nagpal, A. Jatain, and D. Gaur. Review based on data clustering algorithms. In 2013 IEEE Conference on Information & Communication Technologies, pages 298–303. IEEE, 2013.

[26] R. Nugent, E. Ayers, and N. Dean. Conditional subspace clustering of skill mastery: Identifying skills that separate students. International Working Group on Educational Data Mining, 2009.

[27] Y. Park and I.-H. Jo. Factors that affect the success of learning analytics dashboards. Educational Technology Research and Development, 67(6):1547–1571, 2019.

[28] V. Podgorelec and S. Kuhar. Taking advantage of education data: Advanced data analysis and reporting in virtual learning environments. Elektronika ir Elektrotechnika, 114(8):111–116, 2011.

[29] A. Poulos and M. J. Mahony. Effectiveness of feedback: The students' perspective. Assessment & Evaluation in Higher Education, 33(2):143–154, 2008.

[30] J. Preece, H. Sharp, and Y. Rogers. Interaction design: beyond human-computer interaction. John Wiley & Sons, 2015.

[31] C. Romero and S. Ventura. Educational data mining: a review of the state of the art. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 40(6):601–618, 2010.

[32] C. Romero and S. Ventura. Educational data mining and learning analytics: An updated survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(3):e1355, 2020.

[33] P. J. Rousseeuw and L. Kaufman. Finding groups in data. Hoboken: Wiley Online Library, 1, 1990.

[34] B. A. Schwendimann, M. J. Rodriguez-Triana, A. Vozniuk, L. P. Prieto, M. S. Boroujeni, A. Holzer, D. Gillet, and P. Dillenbourg. Perceiving learning at a glance: A systematic literature review of learning dashboard research. IEEE Transactions on Learning Technologies, 10(1):30–41, 2016.

[35] S. Shemwell. Futuristic decision-making. Executive Briefing Business Value from, 2005.

[36] B. Shneiderman and C. Plaisant. Designing the user interface: strategies for effective human-computer interaction. Pearson Education India, 2010.

[37] P.-N. Tan, M. Steinbach, and V. Kumar. Introduction to data mining. Pearson Education India, 2016.

[38] K. Verbert, E. Duval, J. Klerkx, S. Govaerts, and J. L. Santos. Learning analytics dashboard applications. American Behavioral Scientist, 57(10):1500–1509, 2013.

[39] A. Weller. Challenges for transparency. arXiv preprint arXiv:1708.01870, 2017.

[40] R. Xu and D. Wunsch. Survey of clustering algorithms. IEEE Transactions on neural networks, 16(3):645–678, 2005.

[41] F. M. Zanzotto. Human-in-the-loop artificial intelligence. Journal of Artificial Intelligence Research, 64:243–252, 2019.

Building Interpretable Descriptors for Student Posture Analysis in a Physical Classroom

Lujie Karen Chen
University of Maryland Baltimore County
Baltimore, MD
lujiec@umbc.edu

David Gerritsen
Carnegie Mellon University
Pittsburgh, PA
dgerrits@andrew.cmu.edu

ABSTRACT

This research presents a process for simplifying video labeling and feature generation when building classification systems from real classrooms. Using video from a single, wide-angle recording of a live classroom, we create a low-level feature set of posture primitives built on keypoints from OpenPose. We use that feature set to build a posture recognition model of "natural labels" built from a scripted posture video using the same classroom. This model provides automatic labels for the real classroom data. We then derive a set of interpretable descriptors to characterize student-specific posture pattern dynamics. We show that those descriptors are able to discriminate between subtle differences in learning activities in a real college classroom.

Keywords

classroom analytics, posture analysis, student activity recognition

1. INTRODUCTION

The field of research into classroom sensing technologies and data mining is growing. One goal of this work is to provide automated feedback to instructors about anything from latent states of the students to overt actions by the teacher [15]. The motivation for this work is usually to empower teachers and scaffold instructional development without always relying on human consultants [13].

The promise of this field is high, but so are the costs. Technical staff and software development are all expensive. Additionally, labeling video data in order to derive insights about student interactions is particularly time-consuming and difficult. This study describes an attempt to reduce that cost. We used a freely available posture analysis tool (OpenPose) to produce keypoint data for human postures which we then used to build a generic set of labels for a class of students. Our goal was to simplify both the application and interpretability of data labels.

Lujie Karen Chen and David Gerritsen "Building Interpretable Descriptors for Student Posture Analysis in a Physical Classroom". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 713-717. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

2. RELATED WORK

Emerging technologies for sensing pedagogical events in live classrooms include the detection of overt student behaviors (e.g., hand raising and gaze direction [1]), latent states (attention and engagement [16, 12]), and instructor actions (e.g., questions, activity sequences, gestures, and physical location in the room [3, 7, 11, 14]). Each approach has its own trade-offs in terms of reliability and effort required, but the models all require a dictionary of human-labeled body postures. Data annotation is time-consuming work requiring special expertise. For example, one must choose between coding in real-time [9, 10] or post-hoc [12, 16], and whether or not to use assisted label production [17].

Feature generation is a related but different concern. Education researchers may want to build models on comprehensible features, such as the words used during teachers' questions [3, 14], or the gestures students and teachers exhibit during interactions [5, 4, 7, 12, 16]. While it is possible to use a "kitchen sink" approach to quickly assess the success of an algorithm and its inputs, education researchers may prefer to use features that can be observed and understood by the end-user. This way the instructors using their systems might be able to make changes based on the model output, e.g., [2].

3. MOTIVATION

High-quality video cameras are ubiquitous. Researchers in education and machine learning can quickly generate large volumes of dynamic, rich data from the classroom. When turning video into data, education researchers traditionally code classroom videos using any number of methodologies [8]. Each approach for annotating and interpreting video data takes a significant amount of training and time.

To this end we present the following case study in which we demonstrate a pipeline that requires only minimal resource investment on the part of the experimenter, including the time it would normally take to define, identify, and verify student gestures. We propose that this savings is possible without sacrificing the interpretability of a human-coded feature set. To test the pipeline, we designed an easy and accessible feature generation strategy which we then tested against the most difficult in-class dataset we could imagine.

Figure 1 illustrates our workflow, broken into the following stages: (1) We collect a video recording (scripted posture data, section 4.2) with synchronized, scripted posture pat-

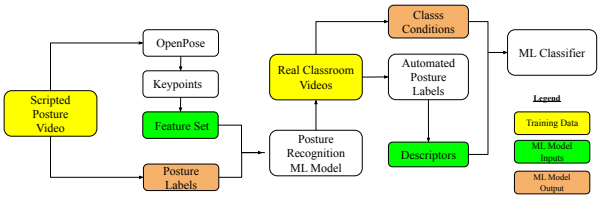


Figure 1: Workflow

terns commonly observed in classrooms; (2) We fit supervised machine learning models to automatically recognize the scripted posture patterns (section 5.1); (3) Using video from a real classroom (section 4.1), we demonstrate the utility of those auto-estimated posture patterns by applying the posture detection models to the real classroom dataset to discriminate between class conditions (section 5.2).

4. DATA SOURCES & AUTO-LABELING

Here we describe our data collection and analysis. We conducted the study at Carnegie Mellon University in the Spring semester of 2019. We generated model data from a group of volunteers, and target data from a class of real students. All students signed IRB-approved consent forms.

4.1 Real Classroom Data

In selecting a use-case for our approach we chose a class that embodied a traditional lecture-based class. We worked with a semester-long graduate level course on "Applied Data Science" (ADS) in Spring 2019. There were 22 enrolled students, all of whom could fit in a single frame of a wide-angle camera (Marshall CV505). The camera faced the rows of students, and the instructor was not in frame. We recorded throughout the entire semester of bi-weekly, 75-minute session. We collected 22 sessions for a total of about 30 hours of class time.

The format of the class was almost completely dominated by professor lecture. Halfway through the semester the students were put into groups for their final projects. During that second half of the semester, student groups took turns giving short presentations throughout the second session of each week. We used this naturally occurring difference in class format to inspire our classification problem. We thus generated two main class conditions: those led by the professor (i.e. *Professor-Lead*), and those led by groups of students giving project presentations (i.e. *Peer-Lead*). In the *Professor-Lead* condition (16 sessions), students listened to the professor lecture and were permitted but never required to ask questions. In the *Peer-Lead* condition (6 sessions), students listened to groups of peers take turns giving a short presentation describing their progress on an ongoing class project. After each presentation, all students were allowed to ask questions, and a random selection of students were required to ask questions for participation points.

Our goal was to model generic student posture patterns as descriptors to discriminate between *Professor-Lead* and *Peer-Lead*. From a naive perspective, the postures of the students in each condition were virtually indistinguishable. We chose this objectively difficult classification problem in order

Figure 2: A snapshot of scripted posture video in which volunteers were performing one of the scripted action of *Checking Phone*

to stress-test our approach. Our proposition was that students in either condition might have different internal states related to their expectation of learning useful information (*Professor-Lead*) vs. their potential requirement to ask a question (*Peer-Lead*), but that we would not have predictions about which gestures might reveal those latent states. This is a "good enough" test of our goal of building a practical process that could eventually be of some potential use to researchers who are likely to test less fuzzy classification problems.

4.2 Generic Student Posture Descriptors

To address our goal of helping researchers create descriptors without deep, costly annotation, we designed an approach that would create a catalogue of possible postures students exhibit in a typical class. We began by creating a 7-minute video of 11 volunteers arranged in the same seats as the students from the ADS class. Using the same equipment as would be used in the real class, we led the volunteers through a series of scripted movements. The "Scripted Posture Video" section of our workflow (Figures 1 and 2) comprises these data. The volunteers did not know what the prompts would be in advance, and their behaviors appeared natural. We guided them through 13 generic posture patterns: *Checking Phone*, *Looking at Computer*, *Looking Down*, *Looking Up*, *Looking at front-left*, *Looking at front-right*, *Looking Left*, *Looking Right*, *Performing Q&A*, *Talking to neighbor*, *Raising hand* (left and right) and *Writing*. We chose this list as a comprehensive representation of observable posture patterns in real classrooms when students listen to lectures. There are additional gestures we could have included, such as sleeping, eating, or drinking. However, this seemed outside of the scope of training on only the most frequent and probable behaviors rather than trying to include every conceivable movement that might exist.

Our next step was to produce an underlying set of low-level features for defining these generic posture patterns. Table 1 is a partial list of the 24 frame-by-frame features we created from OpenPose keypoints [6]. OpenPose[1] is a freely available toolkit for identifying physical landmarks, or "keypoints," on human figures in a picture, as shown in Figure 3 (left). Each keypoint is part of a 2-D array of real-valued numbers (Figure 3, right plot). In this analysis we only use upper body keypoints, including head, neck, and arms.

[1]https://github.com/CMU-Perceptual-Computing-Lab/openpose

Feature Name	Description
neck_nose	neck nose distance
Lshoulder_nose	left shoulder nose distance
Rshoulder_nose	right shoulder nose distance
rHand_nose	right hand nose distance
lHand_nose	left hand nose distance
nose_neck_h	nose neck horizontal displacement
nose_neck_v	nose neck vertical displacement
nose_shoulder angle	nose neck angle w.r.t shoulder
rElbow_angle	right elbow angle
lElbow_angle	left elbow angle

Table 1: A partial list of low-level features used in the posture recognition machine learning model

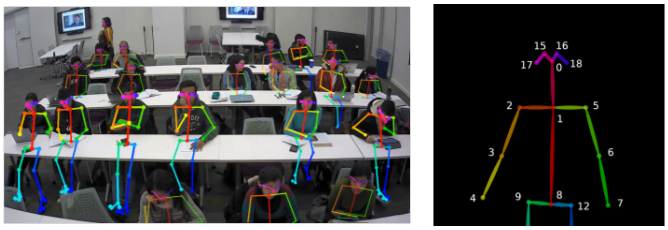


Figure 3: An example of OpenPose toolkit keypoints for a given frame of real classroom data (left); and the upper body keypoints used in our analysis (right)

We designed the features from Table 1 based on our professional experience performing student observation and an analysis of how groups of keypoints move together to produce gross postures. For example, the features *rHand_nose* and *lHand_nose* each measure the vertical distances between the nose and the hand. These distances can indicate vertical hand movements, e.g., as seen in hand-raising. "Nose-neck" related features (e.g. *nose_neck_h* or *nose_neck_v*) can indicate left/right head movements. With these low-level features in hand we then applied them to the scripted posture data and train a random forest classifier for recognizing posture patterns. We compiled a training set with each data point representing a person-frame pair and labeled each data point with labels naturally available from scripted posture data. We then fit several independent binary classifiers, each predicting the binary label of whether a given posture pattern occurred.

Our hope was that by having 11 different people perform the scripted movements in their own unique fashion, the model would be exposed to a sufficient amount of variability—such as one would expect to see in a the real world. We worked from the assumption that this would at least reduce the need for building and applying a precise annotation manual. This allowed us to quickly compile a posture-recognition model.

5. RESULTS

In this section, we present the frame-by-frame posture recognition model (section 5.1) using our generic behavior labels from the scripted posture dataset (4.2). We then applied that model to the ADS dataset (4.1), automatically labeling the posture patterns frame-by-frame. We derive descriptors from each 5-min segment of classroom video based on those machine labels, and built a classifier to discriminate between the two class conditions. The *Peer-Lead* segments of video did *not* include periods of question-asking after student presentations. This was meant to maximize surface similarity between the two conditions and provide a challenging test.

5.1 Posture Patterns Recognition

Table 2 summarizes the Area Under Curves scores (AUCs) for binary classifiers predicting whether a given posture pattern occurred in the appropriate position. An AUC rating of 0.50 is equivalent to chance, which means that *check-phone*, for example, does not have a reliable posture pattern as a composition of its keypoint structures. However, the model is able to identify head movement in left, right and up directions somewhat more reliably than other types of subtle head movements, such as *look-down*, *look-front-left* or *look-front-right*. Hand-raising postures and *writing* are also found to be relatively easier to identify. Similar to *check-phone*, actions without clear movement patterns exhibit low performance, i.e., *look-at-computer*, *Q&A*, and *talking-to-partner*.

Posture Patterns	AUC Scores	Posture Patterns	AUC Scores
check-phone	0.51	look-up	0.80
look-at-computer	0.63	look-left	0.84
look-down	0.53	look-right	0.88
look-front-left	0.67	writing	0.81
look-front-right	0.75	raise-left-hand	0.81
Q and A	0.63	raise-right-hand	0.84
talk-to-partner	0.60		

Table 2: Area Under Curve (AUC) scores from binary classifiers each predicting whether or not a given posture pattern has occurred, from leave-one-person out cross-validation experiment.

5.2 Discriminating Class Conditions

In this section we report the results from testing the hypothesis that there are discernible differences in students' posture patterns between the *Professor-Lead* and *Peer-Lead* class conditions. To answer this question, we formulated a machine learning classification task in which we used the descriptors derived from videos of the ADS class as input (right part of Figure 1). For output labels we used the class conditions. In creating the training dataset, we extracted a series of non-overlapping 5-minute segments from the real classroom videos and computed a list of statistics based on predicted student-by-student, frame-by-frame probabilities of posture patterns from the generic behavior model described in section 4.2. For each 5-minute video segment we derived five statistical values (*mean*, *standard deviation*, *min*,*max*, and *median*) summarizing the predicted probabilities of each of the 13 posture patterns. As a result, we have a training dataset with 65 features (13 posture patterns by 5 statistics) with each row representing a student-segment pair. We use random forest to fit the model. For comparison, we also derived an independent set of low-level features using only the keypoint structures described in Table 1.

We conducted two types of cross-validation experiments: *random split* and *leave-one-session-out*. In *random split* mode, the training and test datasets were constructed by random selection from the pool of 5-minute video segments,

irrespective of the class sessions to which they belonged. This design can yield relatively optimistic performance because of the likelihood that the segments from the same session can appear both in training and testing. In the second experiment, the split is based on class session, which results in a more conservative measurement of discrimination.

Table 3 shows the AUCs for model discrimination between *Professor Lead* vs *Peer Lead* using two different sets of input variables and under two different experimental conditions. The AUCs for each experimental design is beyond a random chance of 0.5. Specifically, we note that AUC decreases when using model-based descriptors compared to using a low-level, less interpretable feature set derived directly from the key points. The drop of AUC from *random split* to *leave-one-session-out* cross-validation suggests that certain predictive features are session specific and therefore make it difficult to predict labels for an unseen session. Future work will be of interest to identify those session specific features and investigate their roles in predicting class conditions.

Input Variables	Random Split	Leave-One Session Out
Posture-Model-based descriptors	0.72	0.64
Keypoint-based features	0.82	0.68

Table 3: Area Under Curve (AUC) scores for experiments to discriminate between *Professor-Lead* and *Peer-Lead* class conditions, comparing *random split* and *leave-one-session-out* cross-validation designs.

In order to understand the features that contribute to discriminating between the class conditions, we reviewed the feature importance from the random forest model that used interpretable posture-model-based descriptors as well as the low-level keypoint features. Figure 4 shows a selection of important and unimportant features from each approach. As noted in the upper portion of Figure 4, the most important input variables in the posture-based model are those describing the variation of left and right head movements. Other posture patterns, such as looking to the front, raising hands, and looking down did not play an important role in the model. The bottom portion of the figure shows that some of the important features were the distances between students' eyes and the angles between their nose and shoulders. Some of the less important keypoint structures included the relative angles of the left shoulder and elbow, as well as the distance between the left shoulder and the nose.

6. DISCUSSION

In this project we explored methods for extracting posture-related descriptors from videos of students in a real classroom. We derived the posture labeling model from a video of volunteers following scripted prompts. We extracted keypoint data from the videos using OpenPose, a freely available general purpose posture keypoints detection tool. We then showed that this method of automatic labeling could distinguish between two highly similar class conditions.

Large body movements such as hand raising and left/right head shifting were the easiest for the model to detect, and the most important descriptors in the posture model. In terms of using labels that are easy to interpret, these types

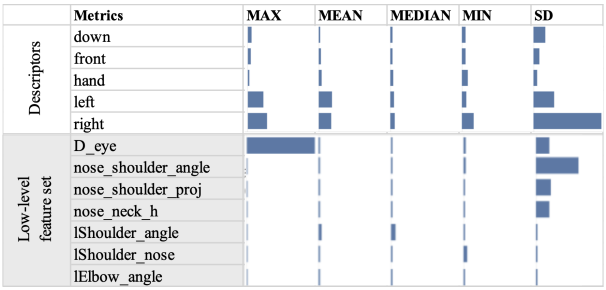


Figure 4: A selection of posture-based descriptors (white) and low-level features (gray) and their importance in a Random Forest model for discriminating between class conditions.

of movements seem like a promising start. Without trying to interpret those movements at this time, they were at least important to the posture model. It maybe the case that simply informing an instructor about these movements could be a productive starting point for reflection.

Given that there were a number of features that did not contribute to the models, and that the raw keypoint model performed better than the derived model, we note that there is a trade off between the accuracy of this approach on the one hand, and its interpretability and transferability on the other. When we look at the variance in Figure 4, we see indications that the importance of some features (and the *lack of importance* of others) is more interpretable than the power of different keypoint angles and vectors. These higher level features say something about what students do differently in different scenarios. Our point here is not to deduce what those meanings are, but to show some of the student behaviors that are worth noticing. In terms of transferability, the fact that we built these labels from a 7-minute session of non-student volunteers shows that this approach may have some potential as a one-to-many label generation method, at least when the volunteers use the same classroom as the target students.

Finally, we propose that the pipeline we explored in this project, from feature generation to auto-labeling and from data prepossessing to feature extraction, can all be generalized to other teaching and learning scenarios in physical classrooms. For researchers in this space, i.e., developing classroom-based technologies for sensing behavior and providing automated feedback, our study may help simplify and accelerate their work by simplifying annotation and anticipating features that the end-user can understand.

7. ACKNOWLEDGMENTS

The research reported here was supported, in whole or in part, by the Institute of Education Sciences, U.S. Department of Education, through grant R305B150008 to Carnegie Mellon University. The opinions expressed are those of the authors and do not represent the views of the Institute or the U.S. Department of Education.

8. REFERENCES

[1] K. Ahuja, Y. Agarwal, D. Kim, F. Xhakaj, V. Varga, A. Xie, S. Zhang, J. E. Townsend, C. Harrison, and A. Ogan. EduSense: Practical Classroom Sensing at Scale. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(3):1–26, 2019.

[2] R. S. Baker and K. Yacef. The state of educational data mining in 2009: A review and future visions. *JEDM| Journal of Educational Data Mining*, 1(1):3–17, 2009.

[3] N. Blanchard, P. Donnelly, A. M. Olney, S. Borhan, B. Ward, X. Sun, S. Kelly, M. Nystrand, and S. K. D'Mello. Identifying Teacher Questions Using Automatic Speech Recognition in Classrooms. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 191–201, 2016.

[4] P. Blikstein and M. Worsley. Multimodal Learning Analytics and Education Data Mining: Using Computational Technologies to Measure Complex Learning Tasks. *Journal of Learning Analytics*, 3(2):220–238, 2016.

[5] N. Bosch, S. K. D'Mello, J. Ocumpaugh, R. S. Baker, and V. Shute. Using Video to Automatically Detect Learner Affect in Computer-Enabled Classrooms. *ACM Transactions on Interactive Intelligent Systems*, 6(2):17, 2016.

[6] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[7] J. H. Correa, D. Farsani, and R. Araya. An application of machine learning and image processing to automatically detect teachers' gestures. In *International Conference on Computational Collective Intelligence*, pages 516–528. Springer, 2020.

[8] J. T. DeCuir-Gunby, P. L. Marshall, and A. W. McCulloch. Using mixed methods to analyze video data: A mathematics teacher professional development example. *Journal of mixed methods research*, 6(3):199–216, 2012.

[9] J. M. Girard. Carma: Software for continuous affect rating and media annotation. *Journal of Open Research Software*, 2(1), 2014.

[10] P. Goldberg, Ö. Sümer, K. Stürmer, W. Wagner, R. Göllner, P. Gerjets, E. Kasneci, and U. Trautwein. Attentive or not? toward a machine learning approach to assessing students' visible engagement in classroom instruction. *Educational Psychology Review*, pages 1–23, 2019.

[11] R. Martinez-Maldonado. "I Spent More Time with that Team". In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, pages 21–25, Tempe, AZ, mar 2019. ACM.

[12] B. Ngoc Anh, N. Tung Son, P. Truong Lam, P. Le Chi, N. Huu Tuan, N. Cong Dat, N. Huu Trung, M. Umar Aftab, T. Van Dinh, et al. A computer-vision based application for student behavior monitoring in classroom. *Applied Sciences*, 9(22):4729, 2019.

[13] A. Ogan. Reframing classroom sensing: Promise and peril. *Interactions*, 26(6):26–32, 2019.

[14] L. P. Prieto, K. Sharma, Kidzinski, M. J. Rodríguez-Triana, and P. Dillenbourg. Multimodal teaching analytics: Automated extraction of orchestration graphs from wearable sensor data. *Journal of Computer Assisted Learning*, 34(2):193–203, 2018.

[15] M. K. Saini and N. Goel. How smart are smart classrooms? A review of smart classroom technologies. *ACM Computing Surveys*, 52(6), 2019.

[16] J. Zaletelj. Estimation of students' attention in the classroom from kinect features. In *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis*, pages 220–224. IEEE, 2017.

[17] T. Zhang, C. Xu, G. Zhu, S. Liu, and H. Lu. A generic framework for video annotation via semi-supervised learning. *IEEE Transactions on Multimedia*, 14(4 PART 2):1206–1219, 2012.

Using Data Quality to compare the Prediction Accuracy based on diverse annotated Tutor Scorings

Sylvio Rüdian
Humboldt-Universität zu Berlin,
Weizenbaum Institute
Berlin, Germany
ruediasy@informatik.hu-berlin.de

Niels Pinkwart
Humboldt-Universität zu Berlin,
Weizenbaum Institute
Berlin, Germany
niels.pinkwart@hu-berlin.de

ABSTRACT

Cross-validation is a wide-spread approach to understand how well a prediction model performs with unseen data. While this is the state of the art, machine learning is often used for educational purposes in educational data mining. Whether a system is applicable and generalizable in practical settings is based on the cross-validation accuracy. One major problem is that the quality of annotated data is often worse due to different raters that score equal tasks differently, even if they were trained before. In this paper, we did an experiment where 1.200 texts of three difficulty levels in an open writing task for language learning were scored by two tutors independently to get the inter-rater reliability score for measuring the similarity across their grades. We used the existing scorings of other tutors of the system to train a random forest regressor for predicting scorings based on the texts. We found out that the accuracy has a strong relationship to the inter-rater reliability score and propose a new measurement that combines both metrics for scenarios where data was annotated by tutors, that could principally be diverse.

Keywords

Tutoring systems, scorings, data labeling, inter-rater reliability, cross-validation

1. INTRODUCTION

As long as tutor scorings are used as a basis to train machine learning systems, there is a bias of subjectivity. Research has shown that the agreement among scores given by tutors often varies [1]. Depending on the task and scale, tutors reach different inter-rater reliability scores. Practical settings have shown that even when teachers were trained for grading, there is a gap. Thus, formal exams are often graded twice and in case that there is a huge gap, a third grader needs to be taken into account. For the field of machine learning, we need thousands of scored tasks, e.g. for automated essay grading. From the practical point, it is understandable that scorings cannot be done by the same tutor all the time. Tutors' time is a limited resource and thus there is the need to score tasks by different experts. If we consider machine learning approaches, there are many examples of prediction tasks, where researchers try to imitate teacher scorings, based on different features. As the reduction of a text or task to features removes information that could be important for a good evaluation, automatic scorings cannot be perfect. Using data gathered by tutors where even scorings for the same texts or tasks are not always equal we think, that it is not fair to compare the prediction accuracy in education in general if we use tutors' labeled datasets.

In machine learning, the proper way to decide whether a system generalizes well is to do cross-validation [2]. Therefore, the data is split into several pieces. The model will be trained on all the data, except from one piece. This piece is used to evaluate the model as we know features and the concrete label. Based on the features, the system creates a prediction using the trained model. The predicted label can be compared with the known one. With every piece, the leave-one-out method (or alternative ones) can be applied to get an averaged accuracy. The main advantage of this method is to create a prediction on previously unseen data. Thus the evaluation shows whether a model generalizes well. Observing this value in detail, we often notice that the accuracies are between 0.6 and 0.8, e.g. 0.6 for 8 classes and 0.78 for 4 classes in [3] or 0.7 in for 4 classes in [4]. From the perspective of machine learning, these are bad values as it means that 3-4 of 10 predictions are wrong.

To have a good and fair measurement for comparison it is necessary to take the inter-rater reliability of human raters into account as in general, the prediction cannot be better than the ratings among raters that have been used for training the machine. The inter-rater reliability is a score of consistency among raters. According to McGraw & Wong [5], the minimum value should be 0.6 as the cut-off for acceptability. Wang & Michelle [6] did a comparative study to compare human essay scoring and reached an inter-rater reliability score (IR score) of 0.62, using the Intraclass Correlation Coefficient. Williamson proposes that an IR score lower than 0.7 is not applicable [7].

The accuracy of predictions is often measured as the comparison of the prediction of the machine and the rater annotations. But the machine itself was trained based on the raters scores, which could differ among raters [8] [1]. It is not surprising that the predicted scorings by machines correlate with the human rater scorings as they are the training base [6]. In contrast, Williamson has shown statistically significant differences between human and machine rating scores [7]. The question remains: comparing all the systems, what is the best and most applicable one? Using the accuracy only fails as the major problem is the quality of the training data – and not the resulting accuracy in cross-validation.

In this paper, we propose an extension of the cross-validation to have a fair measurement for comparing educational predictions, where training data was gathered from tutors. We focus on language learning and examine two research questions:

Sylvio Rüdian and Niels Pinkwart "Using Data Quality to compare the Prediction Accuracy based on diverse annotated Tutor Scorings". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 718-720. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

RQ1: What is the correlation of the inter-rater reliability score in essay scoring for language learning, compared to the prediction accuracy?

RQ2: Combining the cross-validation with the inter-rater reliability score, what is a fair interpretable measurement taking both metrics into account?

2. METHODOLOGY

To address RQ1, two tutors had the task to score open text submissions of three tasks. All tasks had a different difficulty level, easy (1), medium (2), and difficult (3). For every task, we had 400 user submissions, in sum 1.200. Both tutors got access to the tasks and they got 10 typical scorings for a pre-training. Then, all submissions were scored by both instructors independently of each other, using scores of 1 (very good) to 4 (bad/not acceptable). The scoring procedure lasts 1 week for every tutor.

Then we prepared a random forest regressor [9] as a classifier to train a prediction model for essay scoring based on at least 1.200 scorings for each task, that are already existing in the learning system, independently of the scorings from the previous step. These scorings are created by different tutors, where each text was scored only once. So we did not use the data of the previous step for a comparison to avoid training with the new labeled dataset. From practical settings we know that intermediate grades are quite subjective, thus we concentrate on grades 2 and 3 only, which represent "good" and "satisfactory" that are used as labels for the classification problem. The accuracy for prediction in cross-validation (CV) was gathered for each task separately.

Within the next step, we compared the similarity among both tutors of the first step with the accuracy of the second step to examine a possible relation. Finally, we propose a combination of both metrics that allow a fair comparison of the prediction accuracy with the IR Scores to address RQ2.

3. RESULTS

Figure 1 shows all IR scores and the prediction accuracy in a 10-fold CV. We can see that there is a good correlation between these metrics (correlation 0.88). We used the same approach for all tasks, but the IR-scores vary from 0.45 to 0.74, and the accuracies in 10-fold CV range from 0.47 to 0.64. The results show that the accuracy, as well as the IR-score, vary depending on the task. But, there is a strong positive relationship between the maximal achieved prediction accuracy and the IR score.

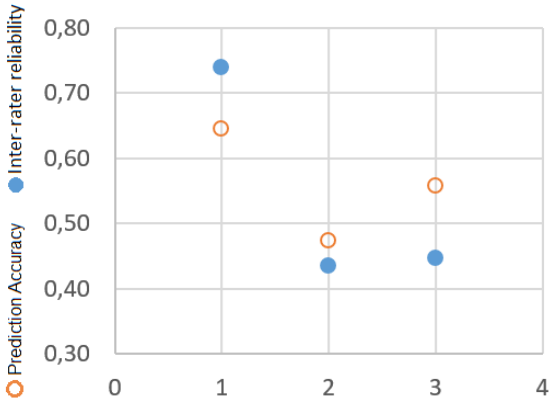


Figure 1. Inter-rater reliability score of two tutors for three tasks, separated by increasing difficulty level and prediction accuracies in cross-validation.

4. NEW MEASUREMENT

The main idea is to combine the classical cross-validation with the inter-rater reliability score. The CV addresses the accuracy of a trained prediction model. As there are multiple versions of the CV, e.g. leave-one-out or leave-p-out (where p is a range of the dataset), we use CV as a general concept and do not limit our approach to a specific version.

The similarity of tutor scorings can be measured by using a correlation coefficient. We chose the Pearson correlation coefficient (PCC) for applying to a sample [10]. It is not outlier resistant [11], but in the area of learning, large gaps can principally occur in ratings, e.g. the score from one rater is "very good / 1" and from another, it is "very bad / 4". This will impact the resulting correlation coefficient. For our new measurement, this is important as this gap influences the training data as well and thus, it influences the prediction accuracy negatively due to a large bias.

We propose a combination of both metrics, namely CV_{PCC}, defined by the following formula:

$$CV_{PCC} = 1 - |CV - PCC^{+}|$$

under the constraint $0 \leq CV, PCC^{+} \leq 1$. The CV accuracy is defined as a number between 0 and 1 [2] and $PCC^{+} = |PCC|$ as we only consider the similarities, not whether the PCC is positive or negative. The CV_{PCC} is a new value where $CV_{PCC} \in [0,1]$, similar to the CV. In the following paragraph, we show that CV_{PCC} cannot be smaller than 0 and never more than 1. Let CV and PCC^{+} be defined as above. Then we examine whether $\exists CV, PCC^{+}$: $1 - |CV - PCC^{+}| < 0$ or $1 - |CV - PCC^{+}| > 1$.

$$1 - |CV - PCC^{+}| < 0 \iff 1 < |CV - PCC^{+}|$$

With $PCC^{+} \geq 0$ we set $PCC^{+} = 0$ to maximize the value for $|CV - PCC^{+}|$. As $0 \leq CV \leq 1$, the maximum value for CV is 1. This follows: $1 < |1 - 0| \iff 1 < 1$, which is a contradiction.

$$1 - |CV - PCC^{+}| > 1$$
$$\iff 1 > 1 + |CV - PCC^{+}|$$
$$\iff 0 > |CV - PCC^{+}|$$

The absolute value x is defined as $\forall x \in \mathbb{R}: |x| \geq 0$ [12]. Thus, with $x = CV - PCC^{+}$: $0 > |x| \iff |x| < 0$. According to the definition of the absolute value, this is not existing in $\mathbb{R}$. Finally, we showed the second contradiction and can conclude that $CV_{PCC} \in [0,1]$. □

In Figure 1 we can see that the CV accuracy, as well as the IR scores, range from 0.44 to 0.74. If we just compare the CV accuracy, we can conclude that there is a high fluctuation. Using the new CV_{PCC}, the scores range from 0.89 to 0.96. Here, the fluctuation is much lower and we now can compare this value with other tools and different datasets.

5. DISCUSSION

In general, we know that having a low inter-rater reliability score is an indicator of a bad quality of training data. Although we used the same amount of data to train the classifier for each task we can observe that it is not fair to compare the achieved accuracy in prediction only. As there is a strong positive relationship between the accuracy and the inter-rater reliability score we propose to combine both metrics when comparing the result with other datasets. Otherwise, that is what our results show, the accuracy differs across tasks, and results are based on the task selection.

In an optimal setting, where all scorings are the same for equal texts across different raters ($PCC^{+} = 1$), follows $CV_{PCC} = \text{CV}$. Observing the other "extreme" side, where the PCC^{+} and the CV values are very low, we can still achieve a high CV_{PCC}, as the

accuracy will be low if labels are diverse for equal feature values. The higher the range between PCC^+ and CV is, the lower CV_{PCC} will be, which means that the relation between both metrics is low. With that information we address RQ2. Thus, this is an indicator of whether the model needs improvement or whether the accuracy cannot become better as the training base has a low quality due to diverse labeling based on different quality expectations of tutors. This interpretation of the value can be helpful to optimize the model. As we use cross-validation as a general metric, our approach is not limited to specific classification methods. We used the random forest regressor, but we can use other classification-based methods like neural networks, support vector machines, or others as long as we get access to the CV score.

We need to emphasize that our method requires a further labeling step to get the inter-rater reliability score across at least two tutors, where each text needs to be labeled twice. This increases the labeling costs. To reduce the amount of work, we could principally use a subset of already labeled texts that has to be labeled by a new tutor to understand the data quality. If a low value will be detected, we know that the resulting accuracy will differ from experiments with other datasets due to the low agreements. We can argue that knowing the problem of diverse scorings is a good fundament to optimize further scorings by a better pre-training of raters. But in praxis, often thousands of labels are existing based on the data that was collected over the last years. Thus, only for future data collection, there can be optimization. If we want to use existing datasets, we propose to use the CV_{PCC} for a fair comparison in relation to other datasets.

In our experiments, we used two separate datasets, one that contains the scorings of the two tutors and one much larger set, where more texts were scored by other tutors. The first was used to get the PCC^+ score and the other to train the classifier based on the maximum achievable CV score. To benefit from the extra labeling, we could enhance the training dataset by the data where the two tutors had equal scorings for the same texts.

Our proposed metric is limited to datasets that were annotated manually. If we have labels that are automatically processed (e.g. the achieved scores in interactive tasks in an online course or whether a student drops out), normally we do not have a diverse annotated dataset. Thus we recommend using the CV_{PCC} in all scenarios where tutors are involved and where diverse annotations (e.g. in scorings) play a role. This is early-stage research, limited to three difficulty levels of specific open-writing tasks. To generalize our findings, the next step is to compare more tasks and the resulting CV_{PCC}. Besides, further studies in other learning domains are required to verify the found relations of the metrics. Our first findings are promising.

6. CONCLUSION

In this study, we examined the relation of the inter-rater reliability of tutor scorings and the accuracy that can be achieved to predict two concrete ratings. In our setting of language learning, we focused on three open writing tasks of different difficulty levels, those accuracies in prediction differ. Based on our results, we observe that there is a strong relationship between both scores, even though both metrics were derived using datasets from multiple raters. Thus we can see that datasets, labeled by tutors, can differ. This infers the data quality and the maximum achievable accuracy in prediction. To use possibly diverse annotated data by tutors and for comparing the prediction results, we propose a new method of combining both metrics to allow fair comparison across different datasets. This new metric can help scientists in educational data mining to compare results of different tutor-based labeled datasets and it helps to understand whether a model or the dataset needs improvement.

7. ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry of Education and Research (BMBF), grant number 16DII127 (Weizenbaum-Institute). The responsibility for the content of this publication remains with the authors.

8. REFERENCES

[1] S. Elliot, "A study of expert scoring, standard human scoring and IntelliMetric scoring accuracy for statewide eighth grade writing responses (RB-726)" Newtown, PA, Vantage Learning, 2002.

[2] J. Sha, "Linear Model Selection by Cross-Validation" in Journal of the American Statistical Association, Taylor & Francis, Ltd, 1993, pp. 486-494.

[3] S. R. Bowman, G. Angeli, C. Potts and C. D. Manning, "A large annotated corpus for learning natural language inference" in arXiv 1508.05326, 2015.

[4] S. Rüdian, J. Quandt, K. Hahn and N. Pinkwart, "Automatic Feedback for Open Writing Tasks: Is this text appropriate for this lecture?" in DELFI 2020 - Die 18. Fachtagung Bildungstechnologien der Gesellschaft für Informatik e.V., 2020, pp. 265-276.

[5] K. McGraw and S. Wong, "Forming inferences about some intraclass correlation coefficients" in Psychological Methods, 1(1), 1996, p. 30–46.

[6] J. Wang and M. S. Brown, "Automated Essay Scoring versus Human Scoring: A Comparative Study" in The Journal of Technology, Learning, and Assessment, 2007.

[7] D. Williamson, "A framework for implementing automated scoring" in Annual Meeting of the American Educational Research Association and the National Council on Measurement in Education, SanDiego, 2009.

[8] P. D. Nichols, "Evidence for the interpretation and use of scores from an Automated Essay Scorer" in American Educational Research Association (AERA), San Diego, CA, 2004.

[9] L. Breimann, "Random Forests" in Machine Learning 45, 2001, p. 5–32.

[10] R. Hunt, "Percent Agreement, Pearson's Correlation, and Kappa as Measures of Inter-examiner Reliability" in Journal of Dental Research, 1986.

[11] Y. Kim, T.-H. Kim and T. Ergün, "The instability of the Pearson correlation coefficient in the presence of coincidental outliers" in Finance Research Letters, Volume 13, Elsevier, 2015, pp. 243-257.

[12] E. H. Moore and H. L. Smith, "A General Theory of Limits" in American Journal of Mathematics Vol. 44, No. 2, The Johns Hopkins University Press, 1922, pp. 102-121.

Towards Difficulty Controllable Selection of Next-Sentence Prediction Questions

Jingrong Feng
Language Technologies Institute
Carnegie Mellon University
jingronf@cs.cmu.edu

Jack Mostow
Robotics Institute
Carnegie Mellon University
mostow@cs.cmu.edu

ABSTRACT

Automatic Question Generation seeks to generate questions about a given text for educational purposes such as testing students' comprehension processes while reading. This paper focuses on the task of predicting the next sentence as a way to exercise and assess a crucial skill that comprehension questions often fail to test, namely relating sentences to the context preceding them. We train a BERT-based model of text coherence to estimate the probability that a given sentence will come next in a story. It achieves 68.4% AUC on a held-out test set, significantly above chance. We define an easiness score as the difference between the estimated probabilities of the next sentence and (the likelier of) two distractors, namely the two subsequent sentences. We evaluate our model on data from Project LISTEN's Reading Tutor by correlating the easiness scores of 1,023 questions against the percentage answered correctly by 274 children. A strong correlation would make it possible to filter such questions by difficulty for children at a specified reading level. Unfortunately, the easiness scores of the questions did not correlate with the correctness of children's answers to them.

Keywords

Automatic question generation, difficulty prediction, next-sentence prediction, reading comprehension assessment, natural language processing, BERT

1. INTRODUCTION

A crucial skill in reading comprehension is inter-sentential processing – integrating meaning across sentences. It involves analysis of cohesive relationships such as coreference, indirect reference, and ellipsis [3]. Inter-sentential processing is hard for young readers partly because it requires assimilation from short-term memory to mid-term memory [12]. Unfortunately, reading comprehension questions often fail to assess inter-sentential information integration [1, 13, 14].

Next-sentence prediction questions are a natural way to test inter-sentential processing and are easy to generate. They are also easy to score, because by definition the correct answer is the next sentence. One form of such questions is true/false, i.e., "Does this sentence come next?" Another form is multiple choice, i.e., "Which sentence comes next?" This form has a higher cognitive load because it requires considering multiple sentences, but may be easier than judging a single candidate sentence by itself. Figure 1 shows both.

Context: Everyone knows that the elephant has a very long nose. But a long time ago, the elephant's nose was short and fat. Like a shoe in the middle of its face.

Does this sentence come next?
− She had a question for every animal.

Which sentence comes next?
− She was curious about everything.
+ One day a baby elephant was born.
− She had a question for every animal.

Figure 1: Two forms of next-sentence prediction questions. Answers in green are correct and answers in red are incorrect.

Although easy to generate and score, next-sentence prediction questions can be hard to answer correctly. For example, one study [2] randomly inserted "Which sentence comes next?" questions in children's stories, with the next three sentences of the story in random order as the choices. Children answered only 41% of these questions correctly, barely above chance and frustratingly low.

Good questions should be challenging but not frustratingly hard. Therefore, difficulty control is important in automatic question generation. However, despite the rapid development of question generation, little work has analyzed the difficulty of automatically generated questions [9], especially for reading comprehension [6, 7, 16], and none of it addresses next-sentence prediction questions.

This paper addresses the difficulty of such questions, and is organized as follows. Section 2 describes how we trained a coherence model to estimate the probability that a sentence comes next given the preceding context, and how we used it to score question easiness. Section 3 evaluates this model on a corpus of children's stories. Section 4 correlates the easiness scores of the questions against the percentage of children who answered the questions correctly. Section 5 concludes.

2. COHERENCE ESTIMATION

To estimate the coherence between a given context and sentence, we fine-tuned a BERT-based binary classification model.

Jingrong Feng and Jack Mostow "Towards Difficulty Controllable Selection of Next-Sentence Prediction Questions". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 721-725. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

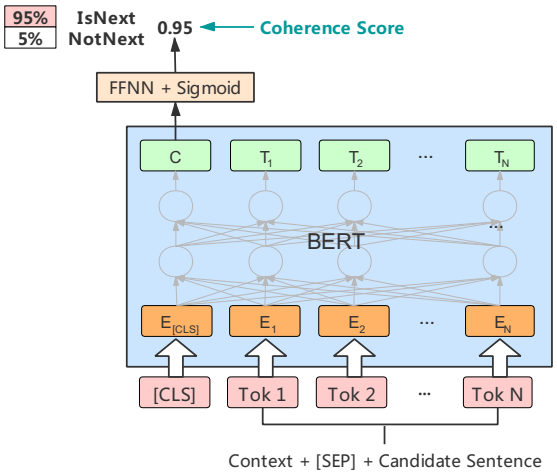


Figure 2: Architecture of the BERT-based model for coherence estimation.

BERT [5], a widely used Transformer-based language model, has achieved state-of-the-art performance on a large suite of natural language processing tasks. The blue box in Figure 2 shows the architecture of the pre-trained BERT model. To do classification, it appends a 2-layer feed-forward neural network (FFNN) to the BERT model, followed by a sigmoid function to scale the FFNN's output between 0 and 1.

BERT was pre-trained on BooksCorpus (800M words) [17] and English Wikipedia (2,500M words) with two objectives. First, randomly masking various words in a text and predicting the masked words from the surrounding text forced BERT to embed each word based on the surrounding words. Second, predicting whether one sentence follows another sentence in the original text forced BERT to learn inter-sentential coherence. Thus these two objectives prompted BERT to learn both intra- and inter-sentential semantic structure.

The effect of the next-sentence prediction task in pre-training has recently been questioned [4, 8, 15]. Some researchers believe that BERT actually learns inter-sentential topic similarity rather than coherence, because its negative instances are sentences sampled randomly from the entire text corpus, which are likely to be topically unrelated to the context.

We now describe how we adapted the BERT-based model to estimate inter-sentential coherence in children's stories.

Input: We fine-tuned the pre-trained BERT-based model on input token sequences of the following form:

- a special token `[CLS]` used for classification tasks
- three sentences of context, which we assume suffice to capture the semantically relevant content. Any more might include irrelevant information or exceed BERT's input length limit of 512 word pieces (i.e., roots and morphemes).
- a special separator token `[SEP]`
- a candidate next sentence; for positive instances, the sentence immediately following the context.

Selection of negative instances: We wanted the task to test children's judgment of inter-sentential coherence, not merely topical relevance. Therefore, rather than sample negative instances randomly from the entire corpus, we selected them from the same story, specifically the 2 sentences immediately following the correct sentence, which are likelier to be topically relevant to the local context than sentences from later in the story. Using the 3 sentences following the context as the multiple choice candidates also matched the task performed by the children in our evaluation dataset, to be described in Section 4.

Human experts could presumably pick contexts and distractors more judiciously to test children's judgements of inter-sentential coherence. However, such manual selection is neither economical nor scalable. One goal of this work was to identify requirements for choosing better contexts and distractors so as to improve automated selection.

Positive-negative ratio of training instances: BERT was pre-trained on equal numbers of positive and negative instances. In contrast, the 3 candidate sentences after the 3-sentence context included one positive instance and two negative instances.

Training labels: To fine-tune BERT and train the FFNN, we set the output of the combined model to 1 for positive instances and 0 for negative instances.

Easiness scores: To measure each candidate sentence's coherence with the given context, we used the probability output by the sigmoid function. Given this measure of coherence, we used a simple heuristic to rate the easiness e of answering a 3-choice question:

$$e = c_{pos} - max(c_{neg_1}, c_{neg_2}) \quad (1)$$

Here c_{pos} is the coherence of the correct answer, and c_{neg_i} is the coherence of distractor neg_i. This formula assumes that the difficulty of the question depends on whichever distractor is more coherent with the context. (As a reviewer suggested, we also tried the log ratio of the two coherence scores instead of their difference, but it performed the same in the evaluation reported in Section 4.)

Figures 4 and 5 show example questions with easiness scores of 0.244 and -0.262, respectively (see Appendix). A negative easiness score occurs when a distractor has greater coherence than the correct answer.

3. EVALUATION OF COHERENCE MODEL

We now evaluate how accurately our coherence model classified the 3 sentences following a 3-sentence context as $IsNext$ or $NotNext$.

3.1 Text Dataset

We constructed a dataset for fine-tuning and evaluating our coherence model from a corpus of English-language children's stories from two sources:

Table 1: Examples of Cases Removed by Data Cleaning

Type	Context	Correct Answer	Choices
two identical choices	...Did the frog slip?	Yes.	<Yes.> <Yes.> <The frog swam fast.>
one choice appearing in the context	...Yes.	Yes.	<Yes.> <The frog swam fast.> <It went past Pat.>
very short context	Pop can twist and bend. Pop slips! Pop stops.	Pop sits.	<Pop sits.> <Pat slaps Pop's hand.> <Pop must rub his feet!>
unfinished sentence in the context/content about phonics instruction	real meat peak	What sound do the letters e a make in the words real, meat, and peak?	<What sound do the letters e a make in the words real, meat, and peak?> <near> <leap>

- 337 stories from Project LISTEN's Reading Tutor [2], totalling 39K words with a vocabulary of 8K distinct words, at grade levels K-7.
- 354 stories from www.africanstorybook.org totalling 91K words with a vocabulary size of 11K, with page lengths ranging from one word to multiple paragraphs.

For fine-tuning and evaluation, we split the 337 LISTEN stories into three subsets, with 60% for training, 20% for hyper-parameter tuning, and 20% for testing, so as to ensure that stories in the test set were not seen during training. We used the African Storybook stories to augment the training set.

For every story in the corpus, we used a 6-sentence sliding window to generate next-sentence prediction items of the form ([`3-sentence context`; `IsNext sentence`; `NotNext sentence`; `NotNext sentence`]), with the correct (IsNext) sentence and two (NotNext) distractors to be presented in random order.

To clean the data, we filtered out several cases (illustrated in Table 1):

- **Cases with two identical choices or a choice appearing in the context:** typically caused by repeated sentences in a conversation.
- **Cases with context or a candidate sentence exceeding 125 words:** might cause the input sequence to exceed BERT's input length limit of 512 word pieces.
- **Cases with very short context:** typically caused by short sentences in a conversation that provide too little information to predict which sentence belongs next.
- **Cases with an unfinished sentence in the context:** for some poems or phonics instructions, sentences were not segmented according to sentence separators.
- **Cases about pronunciation or spelling:** are not relevant to semantic coherence.
- **Cases with the same context followed by different sentences:** may confuse the model during training.

As a result, we got a dataset consisting of 10,761 instances for training, a development set of 1,716 instances for hyper-parameter tuning, and a test set of 2,340 instances for evaluation.

3.2 Training

To fine-tune our coherence model, we used BERT$_{base}$ [5] as the backbone, and the AdamW optimizer [10] with a initial learning rate of 1e-3 and a ReduceLROnPlateau scheduler[1]. We used a ReLU [11] activation in the hidden layer of the FFNN, and set the dropout probability of this hidden layer to 0.5. We trained the model with a standard binary cross-entropy loss function weighted by the positive-negative sample ratio of 1:2.

In contrast to pre-training BERT's hundreds of millions of parameters from scratch, fine-tuning the BERT-based coherence model was inexpensive. It took only about 5 minutes on a single Tesla-V100 GPU to optimize the parameters on the training set.

3.3 Evaluation Results

Table 2 evaluates the coherence model on the development and test sets using various metrics: accuracy, weighted-average precision, recall and F1-score, and area under the ROC curve (AUC). To evaluate metrics other than AUC, we set the classification threshold to 0.5 and compared the predicted label with the ground truth label. In other words, we classified an instance as $IsNext$ if the output probability (coherence score) exceeded this threshold, otherwise as $NotNext$. AUC measures the entire area beneath the ROC curve, which plots true positive rate vs. false positive rate at different classification thresholds. AUC evaluates the overall performance of a classification model by aggregating across all possible classification thresholds.

Table 2: Evaluation of the Coherence Model

Dataset	Accuracy	Precision	Recall	F1-score	AUC
Dev	0.608	0.663	0.608	0.620	0.662
Test	0.609	0.679	0.609	0.619	0.684

4. EVALUATION ON CHILDREN'S DATA

We evaluated our easiness scores by correlating them against 274 children's performance on next-sentence prediction questions. These questions were inserted randomly by the spring 2003 version of Project LISTEN's Reading Tutor into 179 English-language stories ranging from grades 3-7. None of these stories were in the dataset used to train the coherence measure used to score easiness. The questions asked "Which will come next?" and presented the next three story

[1] https://pytorch.org/docs/stable/optim.htmltorch.optim.lr_scheduler.ReduceLROnPlateau

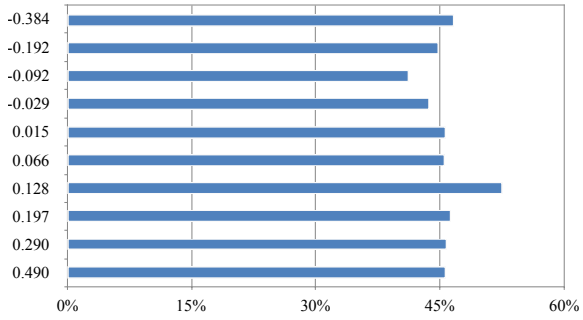


Figure 3: Percentage correct binned by easiness score.

sentences in random order. After data cleaning, we got 1,023 distinct questions with 1,626 responses, of which 45.7% were correct.

344 of these questions had choices with differences in capitalization, as illustrated in Figure 6 (see Appendix). Children might conceivably have used these differences as a clue to eliminate incorrect choices. However, their 622 responses to choices capitalized differently had virtually the same (in fact slightly *lower*) percentage correct (45.5%) as their 1004 responses to choices capitalized the same (45.9%). Evidently children did not make use of this clue. Accordingly, we did not exclude these 622 responses from our dataset.

The questions averaged only 1.59 responses each, far too few to reliably estimate the percentage correct for individual questions. Instead, we split questions by easiness scores into N bins with equal numbers of questions. For N=10, % correct ranged from 41.2% to 52.5%. Figure 3 shows a bar chart with a bar for each of the 10 bins, its average easiness score to its left, and its % correct as its width. The % correct was similar across all 10 bins and unrelated to easiness score. We tried various values of N, ranging from 3 to 128 questions. For each value of N, we correlated the average easiness score of the questions in each bin against their percentage of correct responses. The correlations got weaker as N increased, and were not statistically significant.

To explore why, we regressed response correctness against several features of questions, namely the length and contextual coherence of the correct answer and the two distractors, the length (in characters) of the context, the position of the question in the story (the number of sentences preceding it), and the grade level of the story. We normalized the value of each feature x as $(x - x\_min)/(x\_max - x\_min)$. We performed logistic regression with the normalized feature values for each question as numerical inputs and the correctness of the child's response as binary output. None of the regression coefficients differed significantly from zero. However, their general pattern makes qualitative sense. The contextual coherence of the correct answer was the strongest positive predictor, which makes sense because it measures how well the answer fit the context. The coherence of the harder distractor was the strongest negative predictor, which makes sense because it measures how well that distractor fit the context. The length of the correct answer and the number of preceding sentences in the story were positive predictors, which makes sense because they measure the amount of information provided for selecting the correct answer. Context length and the grade level of the story were negative predictors, which makes sense because reading longer sentences and higher level stories was harder (though better readers read harder stories).

5. CONCLUSIONS

This paper addresses two hypotheses regarding the use of next-sentence prediction questions in assessing children's inter-sentential processing during reading comprehension.

Hypothesis 1: An automated measure of text coherence can predict which of the next 3 sentences will come first. To test hypothesis 1, we trained a BERT-based model of a sentence's coherence with the preceding context to predict whether it comes next. It achieved 61% accuracy on a held-out test set.

Hypothesis 2: An easiness metric based on this measure can predict children's accuracy in selecting the next sentence. To test hypothesis 2, we scored the easiness of the 3-way choice as the coherence of the correct next sentence minus the coherence of the strongest competitor. We then related this score to children's performance on 1,023 such questions presented by Project LISTEN's Reading Tutor to the children while they were using it. There was virtually no correlation. Children answered approximately 45% of the questions correctly regardless of their easiness scores or whether the BERT-based model answered them correctly.

5.1 Limitations and Future Work

If hypothesis 2 were true, we could use a BERT-based coherence model to estimate the difficulty of deciding whether a given sentence will come next in a story context. We could then control question difficulty by using this estimate to help decide which sentence prediction questions to ask. Unfortunately, our results did not support hypothesis 2, which raises the issue of why they did not. The predictor coefficients in our regression analysis to explore this issue made qualitative sense but were not statistically significant.

Perhaps children's performance was affected by the added memory load of considering three sentences as choices. Future work could kid-test the simpler question "Is this next?".

Another possibility is that our coherence model was too impoverished to reflect children's inter-sentential processing. A richer model could capture other aspects such as causal relations, world knowledge, and inference important in story understanding. Or perhaps our BERT model merely needed better adaptation to the domain of children's stories.

An IRT model predicts probability of correctness based on student proficiency minus question difficulty. We did not take direct account of children's differing proficiency, but the Reading Tutor gave children stories at their own reading level, accounting for their proficiency indirectly. Future analyses may need to account for proficiency explicitly.

6. ACKNOWLEDGMENTS

We thank the reviewers for their insightful comments, the children who used Project LISTEN's Reading Tutor, and the team that implemented it and collected our dataset.

7. REFERENCES

[1] J. C. Alderson. Native and nonnative speaker performance on cloze tests. *Language Learning*, 30(1):59–76, 1980.

[2] J. E. Beck, J. Mostow, and J. Bey. Can automated questions scaffold children's reading comprehension? In *International Conference on Intelligent Tutoring Systems*, pages 478–490. Springer, 2004.

[3] N. A. Bond et al. Studies of verbal problem solving: Ii. prediction of performance from sentence-processing scores. technical report no. 87. 1978.

[4] A. Conneau and G. Lample. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[6] Y. Gao, L. Bing, W. Chen, M. Lyu, and I. King. Difficulty controllable generation of reading comprehension questions. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 4968–4974, 2019.

[7] B. S. Hensler and J. Beck. Better student assessing by finding difficulty factors in a fully automated comprehension measure. In *International Conference on Intelligent Tutoring Systems*, pages 21–30. Springer, 2006.

[8] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[9] G. Kurdi, J. Leo, B. Parsia, U. Sattler, and S. Al-Emari. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30(1):121–204, 2020.

[10] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations*, 2019.

[11] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814. Omnipress, 2010.

[12] H. Nomura. Meaning understanding in machine translation. In *Proc. of Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, 1988.

[13] D. Porter. The effect of quantity of context on the ability to make linguistic predictions: A flaw in a measure of general proficiency. *Current developments in language testing*, 5(4):63–74, 1983.

[14] T. Shanahan, M. L. Kamil, and A. W. Tobin. Cloze as a measure of intersentential comprehension. *Reading Research Quarterly*, pages 229–255, 1982.

[15] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[16] C. Y. Yeung, J. S. Lee, and B. K. Tsou. Difficulty-aware distractor generation for gap-fill items. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 159–164, 2019.

[17] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

APPENDIX

Context: George's favorite subject was math. George learned to be a surveyor of land when he grew up. He joined the army and was a leader during the American Revolution.

Choices	Coherence	Easiness
Correct Answer: He later became the first President of the United States.	0.714	
Distractor 1: George Washington is called the "Father of our Country."	0.470	0.244
Distractor 2: We celebrate his birthday on President's Day in February.	0.310	

Figure 4: A question with easiness score of 0.244.

Context: Both Brad and Sally pointed their flashlights into the dark. All they saw were some spider webs and a dead end. The cave was empty.

Choices	Coherence	Easiness
Correct Answer: Brad felt sad.	0.337	
Distractor 1: He had hoped they would find a big pirate ship or something neat.	0.101	-0.262
Distractor 2: Sally looked around the walls of the cave.	0.599	

Figure 5: A question with easiness score of -0.262.

Context: When all the straw was spun away, and all the bobbins were full of gold. As soon as the sun rose the King came and when he perceived the gold he was astonished and delighted.

Choices	Coherence	Easiness
Correct Answer: But his heart only lusted more than ever after the precious metal.	0.695	
Distractor 1: He had the miller's daughter put into another room full of straw,	0.248	0.447
Distractor 2: much bigger than the first, and bade her, if she valued her life,	0.094	

Figure 6: A question with choices capitalized differently.

Sex-Related Behavioral Differences in Online Math Classes: An Epistemic Network Analysis

Yufei Gu
New York University Abu Dhabi
yg1262@nyu.edu

Kun Xu
Spark EdTech
xukun04@huohua.cn

ABSTRACT

The aim of the present study was to examine the rarely studied existence of sex-related behavior difference in online mathematics classes in China. Epistemic Network Analysis (ENA) was utilized in this study to explore the connection of students' classroom behaviors, and the differences in connection patterns for boys and girls. The class monitoring videos of a sample of 64 students (32 male, 32 female) was coded for microscopic categories of in-class behaviors, and all the codes were organized in the format of adjacent matrix. ENA model showed significant results that girls were more likely to engage in social activities in class, while boys exhibited more disruptive behaviors. There was also a relatively stronger connection between disruptive behaviors and call-out behaviors, and a slightly stronger connection between off-task behaviors and disruptive behaviors, and between disruptive behaviors and direct-no volunteer interactions for boys, compared with girls. This study provided an insight into the connection of different categories of classroom behaviors varied by gender, implying a future direction to examine the relationship between different behavioral connection patterns and students' math achievement in online math classes.

Keywords

sex-related behavioral difference, math achievement, teacher-student interaction, Epistemic Network Analysis (ENA), online math class

1. INTRODUCTION

1.1 Background

Previous studies examining sex-related differences in mathematics performance have reached inconsistent conclusions: while some reported a male advantage in math achievement, other studies only found sex-related differences in certain age groups and certain areas of mathematics ability, or even a female advantage in math exams [5][9][25][15][33].

On the other hand, although various research has been conducted to identify the specific contexts and factors that correlated with the difference in mathematics achievement of female and male students [7][11][14][21], the specific classroom behaviors and level of engagement, which have been linked by previous research to varying levels of mathematics achievement, have rarely been studied [10][26][17][22]. In a study conducted by Hart [13], boys were found to be more involved in public interactions in class with their teachers than girls, and the study indicated significant main effects of gender of students on two sub-categories of public teacher-student interaction: open volunteer interactions, and call-out interactions

1.2 Online Math Classes in China

During the past few years, China has witnessed an explosion of different online education platforms, which provides students with easy access to high-quality learning materials regardless of their geographical location. The bloom of online education also provides researchers with opportunities to conduct observational studies on teacher-student interactions without having to set up a camera or be physically present in the classroom. In this case, online learning platform provides a great opportunity for researchers to examine the behaviors of students of different sex in the online classes without influencing or interrupting how teachers and students behave and interact in class. Thus, the present study utilized the classroom monitoring videos from Spark EdTech, a Chinese K-12 online education platform that aims to cultivate mathematics thinking among mandarin-speaking children, to examine whether sex-related behavioral differences exist in online settings adopting the coding rules used in Hart's framework [13][19][8].

1.3 Epistemic Network Analysis

Epistemic Network Analysis (ENA) is a quantitative ethnographic technique designed to address questions in learning analytics and model the structure of connections in the dataset. Major assumption of ENA includes: 1) a set of meaningful features, which is defined as *codes*, can be identified systematically in the data; 2) the data has local structures, which are referred to as *conversations*; and 3) the way in which *codes* are connected to each other within the *conversations* is an important feature of the data [27][28][29].

ENA models the connections between *codes* by quantifying the co-occurrence of *codes* within *conversations*, generating a weighted network of co-occurrences and visualizations for each unit of analysis in the data accordingly. Since all the networks of units are analyzed simultaneously, ENA could ideally produce a set of networks that can be compared visually and statistically alike. Such a method has been used to not only analyze learning data, but also in other context where structure of connections in the data is meaningful, such as communications among health care team and gaze coordination during collaborative work [31][1]. Having recognized the unique power of ENA in analyzing

Yufei Gu and Kun Xu "Sex-Related Behavioral Differences in Online Math Classes: An Epistemic Network Analysis". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 726-730. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

connections within the data, this study adopted ENA for exploring the connections of students' behaviors in mathematics classes, and the differences between connection patterns for students of different sex.

1.4 Goals

The present study aims to: 1) examine the existence of sex-related behavioral differences in online math classes in China; 2) explore the connections of students' in-class behaviors and engagement in classroom activities; 3) compare both visually and statistically the structure of connections of mathematics classroom behaviors for students of different sex.

2. METHODS

2.1 Participants

In order to control for the influence of class content and teaching style on students' classroom behavior and engagement, 12 math classes taught by 2 teachers (1 male, 1 female) of the same topic were randomly drawn from all Level 6 mathematical thinking classes of Spark EdTech. Each class consisted of 5 or 6 students of different sex, making up a sample of 64 students (32 male, 32 female). Teachers from both sexes were selected in order to control for the potential interaction effect of students' sex and teacher's sex on students' behavior. The average class duration for Teacher 1 was 49.13 (SD = 2.49) minutes, and the average class duration for Teacher 2 was 45.43 (SD = 0.84) minutes. Since students went through placement exams that determined their math ability before being assigned to different levels of classes, it can be assumed that students of the same level have similar level of mathematics ability. Level 6 class was primarily designed for third-grade students around eight years old. In this sample, students have the average age of 7.46 (SD = 1.63) years old.

2.2 Procedure

Class monitoring videos were viewed and coded by an experienced coder based on the definition of different types of classroom behaviors proposed in Hart's study (1989). Each students' classroom behavior and interactions with teacher was viewed and coded individually, following an event sampling or episodic approach, which has been widely used in the field of developmental psychology [16][18]. Microscopic categories of behaviors were coded in order to demonstrate the initiations and responses of the students and the teacher. When a behavior lasted for more than 20 seconds, such a behavior was coded again in order to indicate the continuity of that behavior. All the codes were organized in the format of adjacent matrix (see Table. 1 for an example) required by Epistemic Network Analysis (ENA), a sample of which can be found below. Then ENA will be applied to the data using the ENA Web Tool (Version 1.7.0) [20].

Student ID	Sex	Teacher ID	social activity	call out	open volunteer	off task	disruptive behavior
001	1	teacher 0	0	1	0	0	0
001	1	teacher 0	0	0	0	1	1
002	0	teacher 1	1	0	0	0	0
002	0	teacher 1	0	0	1	0	0

Table 1. Illustration of Coding Sheet

2.3 Measures

Several subcategories of students' public interaction with teachers were identified in Hart's study [13]. Two sub-categories of public teacher-student interaction which were found to be significantly correlated with students' sex were: *open volunteer interaction*, and *call-out* interaction. Meanwhile, since another category of public teacher-child interaction, direct-no volunteer interaction, is pretty common in online classes, we decided to also include it as a type of behavior to be examined in our study.

An *open volunteer* interaction was coded when the student indicated in some way other than by calling out a desire to respond to a teacher question or to initiate a public interaction with the teacher. A *call-out* interaction was coded when a target student called out the answer to a teacher question before the teacher gave permission for that student to respond. A *direct-no volunteer* interaction was coded when the teacher asked a question and requested that a target student answer who had not indicated in some way a desire to answer the question. The students usually indicated a desire to respond by raising a hand or calling out.

In addition, we combined two types of behaviors that indicated a lack of engagement in mathematics activities in class which were found to be differently correlated with mathematics achievements for boys and girls in a study conducted by Peterson and Fennema [24]. *Off-task behaviors* were defined in this study as behaviors that are irrelevant to class activities. *Social activities* were defined in this study as the engagement in an activity in which the content of the activity involved a social topic, socializing or discussion of personal information or problems. Another category of behaviors – *disruptive behaviors*, which boys and girls differ drastically in the classroom settings was also included in our measure [4][6]. *Disruptive behaviors* were coded in this study when a student was engaged in behaviors that were likely to substantially or repeatedly interfere with the conduct and discipline of the class.

3. DATA ANALYSIS AND RESULTS

3.1 Definition of ENA Elements

In the present study, the *units of Analysis* were defined as all lines of data relative to a single value of student' sex subsetted by student ID. For instance, one unit included all the lines that represented the occurrence of each category of behaviors for one single student.

In our ENA model, the following codes, which corresponded to the aforementioned five categories of classroom behaviors, were included: social_activity, direct_no_volunteer, off_task_behavior, open_volunteer and disruptive_behavior.

Conversations were defined as all lines of data related to a single value of Teacher Name. For instance, one *conversation* consisted of all the lines associated with one of the two teachers.

3.2 Procedure of ENA

The ENA algorithm adopts a moving stanza window to generate a network model for each line in the data, showing how *Codes* in the current line are connected to codes that appear within the recent temporal context [30], defined as 4 lines (each line plus the 3 previous lines) within a given conversation. The corresponding networks are aggregated for all lines for each *unit of analysis* in the model. In this model, we aggregated the resulting networks using a binary summation where the networks for a given line reflect the presence or absence of the co-occurrence of each pair of *codes*.

The networks for all *units of analysis* in the present model were normalized before being subjected to a dimensional reduction, in order to account for the different amounts of coded lines of different *units of analysis* in the data. In terms of dimensional reduction, a singular value decomposition was utilized, which produces orthogonal dimensions that maximize the variance explained by each dimension [2][28][31].

3.3 ENA Model

Networks were visualized using network graphs where nodes correspond to the *codes*, and edges reflect the relative frequency of co-occurrence, or strength of connection, between two *codes.* The result is two coordinated representations for each unit of analysis: 1) a plotted point graph, which represents the location of that unit's network in the low-dimensional projected space, and 2) a weighted network graph. The positions of the nodes in the network graph are fixed and determined by an optimization routine minimizing the difference between the plotted points and their corresponding network centroids. Because of this co-registration of network graphs and projected space, the positions of the network graph nodes—and the connections they define—can be used to interpret the dimensions of the projected space and explain the positions of plotted points in the space. Our model had co-registration correlations of 0.92 (Pearson) and 0.92 (Spearman) for the first dimension and co-registration correlations of 0.97 (Pearson) and 0.97 (Spearman) for the second. These measures indicate that there is a strong goodness of fit between the visualization and the original model according to the rule-of-thumb by Shaffer, Collier & Ruis [28].

Mean networks for boys' and girls' behaviors in online math classes were constructed by averaging the connection weights across individual networks, and were compared using network difference graphs. These graphs are calculated by subtracting the weight of each connection in one network from the corresponding connections in another (See Figure 3 for a comparison ENA Model for student behaviors in online math classes by sex).

According to Figure 3, the network centroids for boys and for girls differ along the x-axis. There is a relatively stronger connection between disruptive behavior and call out behaviors in online math classes for boys compared with girls. In addition, there is a slightly stronger connection between off-task behaviors and disruptive behaviors, and between disruptive behaviors and direct-no volunteer interactions for boys, compared with girls.

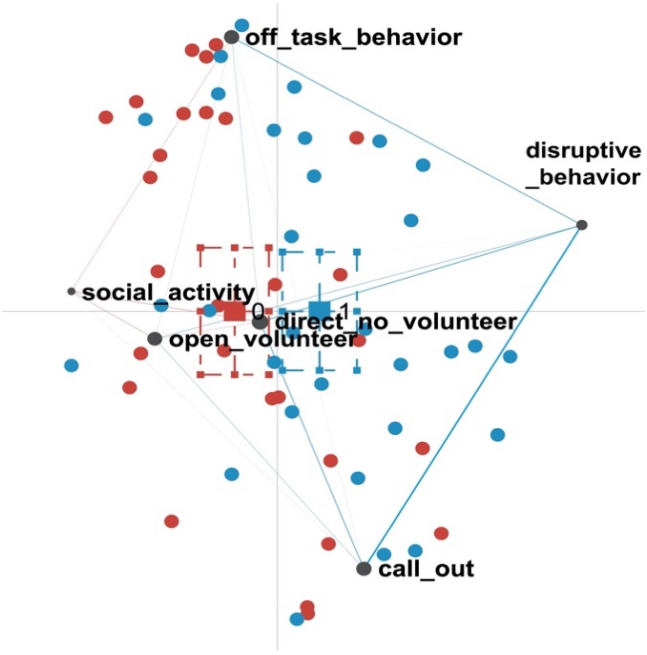


Figure 3. Comparison ENA Model for Student Behavior in Online Math Classes by Sex

*Note: Blue represent boys, red represents girls

4. CONCLUSIONS AND IMPLICATIONS

The present study examined the differences in the structure of connections of classroom behaviors for boys and girls in online mathematics classes in China using Epistemic Network Analysis (ENA). The results indicated a significant difference along the x-axis of the model, suggesting that in our sample, girls were more engaged in social activities and open volunteer interactions with the teacher, while boys exhibited more disruptive behaviors during the class. The difference in off task behavior and call out interaction for boys and girls were not significant. Such a finding is largely consistent with the study conducted by Hart [13], except for we did not find a significant effect of sex on call out interactions. Such an inconsistency might be due to the unique characteristics of online classroom settings, where girls tend to experience higher influence of social presence on their satisfactory level in class, and thus are equally, or even more active in online discussion than boys [32][23].

Thus, in order to elevate their level of satisfaction in online math classes, girls might be more motivated to engage in interactions with their teachers by calling out their answers to teachers' questions than they would normally do in traditional classrooms. Another possible explanation to this phenomenon is the reward system designed by Spark EdTech, a leading online education technology company in China, for its online mathematical thinking classes, where students could receive "little stars" from their teachers by answering questions and participating in classes. Such a reward system could encourage students to participate in classes, and to become the first to respond to the questions by calling out the answers.

Another conclusion we could reach according to the comparison graph is that there is a relatively stronger connection between disruptive behavior and call-out behavior for boys compared with girls. Such a connection indicates that while boys call out their answers to teachers' questions more often, they are more likely to also become disruptive in class by interrupting teacher's lecture or interaction between other students and teacher. It could be implied that boys are more expressive and active in classes, yet such behaviors could become disruptive if they could not regulate their level of activeness in class or if they disregard class disciplines.

In addition, there is a slightly stronger connection between off-task behavior and disruptive behaviors, and between disruptive behaviors and direct-no volunteer interactions. Such connections indicate that boys are more likely to violate discipline in class and be disengaged in class activities at the same time, and teachers are more likely to call on disruptive boys to answer questions, compared with girls. Such findings were consistent with the findings of research conducted in face-to-face classroom settings, that teacher tended to attend more to boys because they were more likely to exhibit disruptive behaviors in class [8][6]. All the findings mentioned above implied the necessity to recognize the

difference in boys' and girls' behavioral patterns in online math classes, and to call for the development of a more gender-sensitive guidance for online teaching, which had been pointed out recently by several researchers [3].

One major limitation of this study is the relatively small sample size. However, since each student's behaviors during a full-length class were coded, we still obtained statistically significant results. At the meantime, due to the time constraint, all the videos were coded by only one coder, which might lead to biased data. The fact that the coder had more than 1000 hours experience of video coding and maintaining an inter-rater reliability of more than 0.9 might, to some extent be able to account for such a limitation. Another limitation of the study arises from the nature of class monitoring videos, which might not always be able to fully capture students' behaviors in class. Scarcely, when students wrote on a notebook or scratch paper, coder was unable to distinguish whether they were taking notes or engaging in off-task behaviors such as sketching. These ambiguous behaviors were not coded, and thus might lead to a slight underrepresentation of students' off-task behaviors.

Overall, the present study is mostly consistent with existing research on sex difference in students' behaviors in traditional in-person classrooms. The novel findings of an insignificant sex-related differences in call-out behaviors could be attributed to the uniqueness of online class settings and the reward system adopted by Spark EdTech. To our knowledge, this study is the first of its kind to employ Epistemic Network Analysis (ENA) to examine the structure of connections of students' classroom behaviors in online math classes in China and conduct a comparison of such connection patterns between boys and girls. Thus, the findings of this study could serve as the first step to examine the relationship between different behavioral patterns in online math classes and math achievement, and to develop a gender-sensitive guidance for online teaching.

5. ACKNOWLEDGMENTS

Our sincere thanks to Spark EdTech for their generous help on providing data for this study.

6. REFERENCES

[1] Andrist, S., Collier, W., Gleicher, M., Mutlu, B., & Shaffer, D. (2015). Look together: Analyzing gaze coordination with epistemic network analysis. *Frontiers in Psychology*, 6(1016).

[2] Arastoopour, G., Swiecki, Z., Chesler, N. C., & Shaffer, D. W. (2015). Epistemic Network Analysis as a tool for engineering design assessment. *Presented at the American Society for Engineering Education*, Seattle, WA.

[3] Asterhan, C. S., Schwarz, B. B., & Gil, J. (2012). Small-group, computer- mediated argumentation in middle-school classrooms: The effects of gender and different types of online teacher guidance. *British Journal of Educational Psychology*, 82(3), 375-397.

[4] Beaman, R., Wheldall, K., & Kemp, C. (2006). Differential teacher attention to boys and girls in the classroom. *Educational review*, 58(3), 339-366.

[5] Benbow, C. P., & Stanley, J. C. (1980). Sex differences in mathematical ability: Fact or artifact?. *Science*, *210*(4475), 1262-1264.

[6] Brophy, J. & Good, T. (1974) *Teacher–student relationships: causes and consequences* (New York, Holt, Rinehart & Winston).

[7] Chipman, S. F., Brush, L. R., & Wilson, D. M. (Eds.). (2014). *Women and mathematics: Balancing the equation.* Psychology Press.

[8] Chiu, M. S., & Whitebread, D. (2011). Taiwanese teachers' implementation of a new 'constructivist mathematics curriculum': How cognitive and affective issues are addressed. *International Journal of Educational Development*, 31(2), 196-206.

[9] Downey, D. B., & Vogt Yuan, A. S. (2005). Sex differences in school performance during high school: Puzzling patterns and possible explanations. *Sociological quarterly*, 46(2), 299-321.

[10] Evertson, C. M., Anderson, C. W., Anderson, L. M., & Brophy, J. E. (1980). Relationships between classroom behaviors and student outcomes in junior high mathematics and English classes. *American educational research journal,* 17(1), 43-60.

[11] Fennema, E. (1984). Girls, women, and mathematics. In E. Fennema & M. J. Ayer (Eds.), *Women and education: Equity or equality?* (pp. 137-164). Berkeley, CA: McCutchan

[12] Grieb, H., & Easley, J. (1984). In Steincamp, M. and ML Maehr (Eds.) *Women in Science*; Volume 2: Advances in Motivation and Achievement," A primary school impediment to mathematical equity; can studies in role-dependent socialization." Greenwich, CT.

[13] Hart, L. E. (1989). Classroom processes, sex of student, and confidence in learning mathematics. *Journal for Research in Mathematics Education*, *20*(3), 242-260.

[14] Hargreaves, M., Homer, M., & Swinnerton, B. (2008). A comparison of performance and attitudes in mathematics amongst the 'gifted'. Are boys better at mathematics or do they just think they are?. *Assessment in Education: Principles, Policy & Practice*, 15(1), 19-38.

[15] Hyde, J. S., Fennema, E., & Lamon, S. J. (1990). Gender differences in mathematics performance: a meta-analysis. *Psychological bulletin*, *107*(2), 139.

[16] Kuczynski, L., & Kochanska, G. (1990). Development of children's noncompliance strategies from toddlerhood to age 5. *Developmental Psychology*, 26(3), 398.

[17] Li-Grining, Christine P., Carolina Maldonado-Carreno, Elizabeth Votruba-Drzal, and Kelly Haas. 2010. "Children's Early Approaches to Learning and Academic Trajectories through Fifth Grade." *Developmental Psychology* 46 (5): 1062–77.

[18] Liu, M., Chen, X., Rubin, K. H., Zheng, S., Cui, L., Li, D., ... & Wang, L. (2005). Autonomy-vs. connectedness-oriented parenting behaviours in Chinese and Canadian mothers. *International Journal of Behavioral Development*, *29*(6), 489-495.

[19] Ma, X. (1999). Gender differences in growth in mathematical skills during secondary grades: A growth model analysis. *Alberta journal of educational research*, 45(4).

[20] Marquart, C. L., Hinojosa, C., Swiecki, Z., Eagan, B., & Shaffer, D. W. (2018). Epistemic Network Analysis (Version

1.7.0) [Software]. Available from http://app.epistemicnetwork.org

[21] Neuville, E., & Croizet, J. C. (2007). Can salience of gender identity impair math performance among 7–8 years old girls? The moderating role of task difficulty. *European Journal of Psychology of Education*, 22(3), 307-316.

[22] Park, S. Y. (2005). Student engagement and classroom variables in improving mathematics achievement. *Asia Pacific Education Review*, 6(1), 87-97.

[23] Park, C., & Kim, D. G. (2020). Exploring the Roles of Social Presence and Gender Difference in Online Learning. *Decision Sciences Journal of Innovative Education*, *18*(2), 291-312.

[24] Peterson, P. L., & Fennema, E. (1985). Effective teaching, student engagement in classroom activities, and sex-related differences in learning mathematics. *American Educational Research Journal*, 22(3), 309-335.

[25] Reis, S. M., & Park, S. (2001). Gender differences in high-achieving students in math and science. *Journal for the Education of the Gifted*, *25*(1), 52-73.

[26] Robinson, K., & Mueller, A. S. (2014). Behavioral engagement in learning and math achievement over kindergarten: A contextual analysis. *American Journal of Education*, 120(3), 325-349.

[27] Shaffer, D. W. (2017). *Quantitative ethnography*. Madison, WI: Cathcart Press.

[28] Shaffer, D. W., Collier, W., & Ruis, A. R. (2016). A tutorial on epistemic network analysis: Analyzing the structure of connections in cognitive, social, and interaction data. *Journal of Learning Analytics*, 3(3), 9–45.

[29] Shaffer, D. W., & Ruis, A. R. (2017). Epistemic network analysis: A worked example of theory-based learning analytics. In C. Lang, G. Siemens, A. F. Wise, & D. Gasevic (Eds.), *Handbook of learning analytics* (pp. 175–187). Society for Learning Analytics Research.

[30] Siebert-Evenstone, A., Arastoopour Irgens, G., Collier, W., Swiecki, Z., Ruis, A. R., & Williamson Shaffer, D. (2017). In Search of Conversational Grain Size: Modelling Semantic Structure Using Moving Stanza Windows. *Journal of Learning Analytics*, 4(3), 123–139. https://doi.org/10.18608/jla.2017.43.7

[31] Sullivan, S., Warner-Hillard, C., Eagan, B., Thompson, R. J., Ruis, A. R., Haines, K., ... & Jung, H. S. (2018). Using epistemic network analysis to identify targets for educational interventions in trauma team communication. *Surgery*, 163(4), 938-943.

[32] Tsai, M. J., Liang, J. C., Hou, H. T., & Tsai, C. C. (2015). Males are not as active as females in online discussion: Gender differences in face-to-face and online discussion strategies. *Australasian Journal of Educational Technology*, *31*(3).

[33] You, Z. (2010). Gender differences in mathematics learning. *School Science and Mathematics,* 110(3), 115-118.

Read & Improve: A Novel Reading Tutoring System

Rebecca Watson
iLexIR Ltd
Cambridge
United Kingdom
bec@ilexir.co.uk

Ekaterina Kochmar
Dept of Computer Science
University of Bath
United Kingdom
ek762@bath.ac.uk

ABSTRACT

We introduce a new readability tutoring system, Read & Improve, a freely available online resource aimed at supporting learners of English and English Language Teaching (ELT) professionals by improving English learners' reading proficiency. Using a combination of machine learning approaches and natural language processing techniques, Read & Improve detects learning needs of every student and makes sure no learner is left behind by identifying reading content at an appropriate level of *readability* and helping learners acquire new words through accessible dictionary definitions and content exploration functionality.[1]

Keywords

Distance Learning, Student Assessment, Natural Language Processing

1. INTRODUCTION

Reading is one of the fundamental language skills. Developing this skill is an essential part of language acquisition, both for native speakers and second language learners [9, 13]. At the same time, developing reading ability takes a considerable amount of time, and, as any learning process, it gets interrupted if readers lose motivation [8, 15]. Such factors as not having a range of engaging reading content offered and being presented with reading material at the wrong level of readability are some of the major contributors to the decreased motivation in readers [11]. In addition to language learners themselves, English Language Teaching (ELT) professionals face similar problems, as finding engaging reading content at the right level of readability is a challenging and a time-consuming task. In this paper, we present Read and Improve (*R&I*), a freely available, open-access educational system that is aimed at both language learners and teachers.[2]

To ensure that the reading content provided to a learner is at an appropriate level of readability, *R&I* uses machine learning methods described in [18] to automatically label texts with readability levels corresponding to the Common European Framework of Reference for Languages (CEFR) [6]. The CEFR is an international standard that describes language ability on a six-point scale from A1 for beginners level up to C2 for advanced level of language proficiency.

To ensure that the reading content presented to a learner is engaging, *R&I* employs news articles that are sourced from news websites in real time. To source news content, *R&I* monitors both RSS Feeds from news websites and the publicly available Common Crawl News (`CC-NEWS`) Dataset.[3] A fully automated *Indexing Pipeline* (RIIP, herein) processes news articles and automatically labels the *readability* of each article's text. News articles are generally available for learners on *R&I* within 10 minutes of publishing on an RSS news feed and in 3-6 hours of the article's publishing time if sourced from `CC-NEWS`. As compared to other domains, news articles have the additional benefit of being generally free of grammatical and spelling errors, which allows us to achieve more reliable linguistic analysis and to provide learners with high quality reading content. *R&I*'s user interface (UI) enables learners to not only read the latest news articles but also to perform keyword search to find articles on topics that they are interested in at their desired CEFR level(s).

A number of applications for various groups of readers, including native and non-native speakers, readers with cognitive impairments, and children, to name just a few, have been developed in recent years. In contrast to the previous work [13, 16, 17], our platform is aimed specifically at developing reading ability in non-native speakers of English. Our approach bears similarities to the Read-X [14] and REAP [10] systems, while also being actively developed and supported as an open-access educational platform available online. *R&I* is markedly different from other available applications, as in addition to providing text search functionality (as in [5]) and vocabulary acquisition help (as in [4]), it supports comprehension testing and personalisation.

[1]This work has been done while the second author was a Senior Research Associate at the University of Cambridge. We thank Cambridge English for supporting this research via the ALTA Institute. We are also grateful to the anonymous reviewers for their valuable feedback.

[2]`https://readandimprove.englishlanguageitutoring.com/`

[3]`http://commoncrawl.org/2016/10/news-dataset-available/`

Rebecca Watson and Ekaterina Kochmar "Read & Improve: A Novel Reading Tutoring System". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 731-735. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

The rest of this paper is structured as follows: Section 2 provides an overview of the system's architecture, Section 3 describes the current UI functionality, and finally Section 4 concludes the paper and describes future work.

2. SYSTEM ARCHITECTURE

Figure 1 illustrates the system architecture of *R&I*. We do not describe the full details of system components here,as this is outside the scope of the paper. Instead, we provide a general overview of the components and their use of natural language processing (NLP).

2.1 API

The API connects to an information retrieval index ('IR Engine'), a database ('DB Engine'), and several APIs to provide the data and search functionality required by the UI. The IR Engine employs Elasticsearch[4] (ES) and includes several distinct indices that facilitate search over news articles and other data.

2.2 RIIP

RIIP is responsible for processing articles into the ES article index. In order to prevent duplicate processing, the pipeline modules first check whether the output file(s) already exist in the 'Data Lake', a single store of all data processed. The API monitors the set of URLs listed in RSS feed(s) and the set of `CC-NEWS` files for new items, and if found, these are sent to RIIP for processing. Therefore, ingestion of new articles through the system requires no manual effort, and up-to-date news content is continuously processed and made available to learners via the UI.

RIIP modules include: the *Extractor*, that extracts text and other information from news articles (i.e. HTML); *RASP*, that parses the text to provide linguistic information [2];[5] the *LevelMarker* module, that labels the text for readability (on the CEFR scale); and finally the *ES* module that indexes text and other linguistic information.

2.3 LevelMarker Module

For RIIP's LevelMarker module we follow Briscoe et al. [3], and define the task of learning readability levels as a discriminative preference ranking task. We employ their machine learning (ML) software and use linguistic features outlined by Xia et al. [18] that represent a text's readability.

2.3.1 Data

We have crawled three publicly available news websites to create datasets: Breaking News English (BNE)[6] (2771 articles), News in Levels (NIL)[7] (6373 articles) and Tween Tribune (TT)[8] (7768 articles). These websites have news articles labelled in terms of their readability however each website's readability levels are based on different scales as shown in Table 1.[9] Each of these datasets are considered to be *parallel* as they contain multiple versions of the same articles simplified across different levels. While the BNE and NIL datasets are designed for L2 English learners, the TT is designed to help L1 learners (early and school-aged readers).

Table 1: Dataset levels and distributions.

(a) BNE

BNE level	CEFR level	Count
0	A2	386
1	A2	386
2	A2	386
3	A2-B1	418
4	B1-B2	392
5	B2	392
6	C1-C2	412

(b) NIL

NIL level	Count
1	2126
2	2124
3	2123

(c) CER

Exam	CEFR level	Count
KET	A2	64
PET	B1	60
FCE	B2	71
CAE	C1	67
CPE	C2	69

(d) TT

TT level	Count
Grade K-4 (0)	1965
Grade 5-6 (1)	2029
Grade 7-8 (2)	1771
Grade 9-12 (3)	2003

Table 2: 5-fold cross-validation tests for each dataset.

Source	Pearson's	Spearman's	Kendall's
BNE	0.8338	0.8368	0.6873
NIL	0.9217	0.9164	0.7880
TT	0.9055	0.9250	0.8071
CER	0.9155	0.9185	0.8015

2.3.2 Evaluation

RIIP employs a model trained on the full BNE dataset as this dataset can be reliably mapped to the CEFR scale (Table 1). Based on this mapping we determined the ranges of ML scores that corresponded to each CEFR level (using observed score range from training data). We tested our model on the Cambridge English Readability (CER) dataset,[10] a publicly available dataset of 331 texts spanning CEFR levels A2 to C2 [18]. On this test set, our model achieves 0.83 Pearson's, 0.85 Spearman's and 0.71 Kendal's correlation coefficient. We also ran 5-fold cross-validation for each dataset[11] and present the results in Table 2.

2.4 ES index

In addition to article index, we create 'WordInfo' and 'CALD' indexes. The CALD indexing system processes definitions from the Cambridge Advanced Learner's Dictionary (CALD) to populate the CALD index. The LexDoop system employs Hadoop[12] to process the Data Lake files (currently around 1 million articles) to produce raw frequency counts of linguistic properties for every word lemma.[13]Following this step, these lemma statistics are collated and added to the 'WordInfo' index.

[4] `https://www.elastic.co/products/elasticsearch`
[5] `https://ilexir.co.uk/rasp/index.html`
[6] `https://breakingnewsenglish.com/`
[7] `https://www.newsinlevels.com/`
[8] `https://www.tweentribune.com/`
[9] BNE to CEFR level map provided by the website: `https://breakingnewsenglish.com/news_levels.html`
[10] `https://ilexir.co.uk/datasets/index.html`
[11] We split the data randomly into training and test sets, ensuring an even distribution of class labels.
[12] Apache Hadoop: `https://hadoop.apache.org/`
[13] LexDoop is also used to process `CC-NEWS` files in parallel.

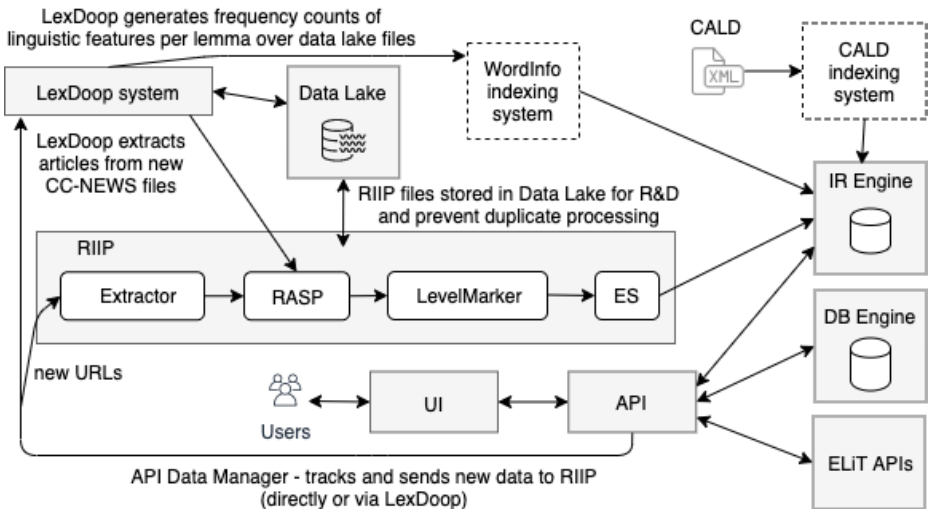


Figure 1: Overview of *R&I* architecture. *R&I* is hosted within, and relies upon, cloud computing services from Amazon Web Services (AWS). Components that use cloud AWS services are shown with grey backgrounds.

2.5 Sanitisation

To make sure the content provided on the platform is acceptable for a wide range of readers across various ages and cultures, we apply content *"sanitisation"* strategy, whereupon we automatically filter out news articles that contain words pertaining to the topics that might be considered offensive in some cultures or inappropriate for younger readers. The list of around 1600 such taboo words was curated using the lists of taboo words from social media. Sanitisation is run within RIIP and the API and, in case the sanitisation system makes an error, the UI enables admin users to mark articles as 'unsafe' (or vice versa).

3. READING ON THE PLATFORM

We define the *R&I* functionality in terms of four major aspects, which cover the tutoring system's ability to provide learners and teachers with engaging reading content at the appropriate level of readability (§3.1); help learners develop their vocabulary in English (§3.2); run comprehension tests (§3.3); and allow learners to revisit texts they read, words they clicked on and tests they submitted (§3.4).

3.1 Finding engaging reading material at an appropriate level

The first step for learners accessing *R&I* is to define their language proficiency level. Learners can log in to *R&I* using their account credentials from Write & Improve,[14] a freely available system linked to the reading platform, that is able to assess and provide feedback on a learner's writing proficiency. Once logged in, *R&I* defaults reading proficiency to current writing proficiency, but a learner can change their CEFR reading level.

Figure 2 contains a screenshot of the *search page*'s results showing the latest news articles at the learner's CEFR level (currently B1). The search page provides learners with snippet(s) of the article text, and they can click on any of the titles listed on this page in order to load the *article view page* where they can read the article itself. In addition, search by keywords is enabled on *R&I* to allow learners to find articles not only at their level of readability, but also on the topics of their interest.

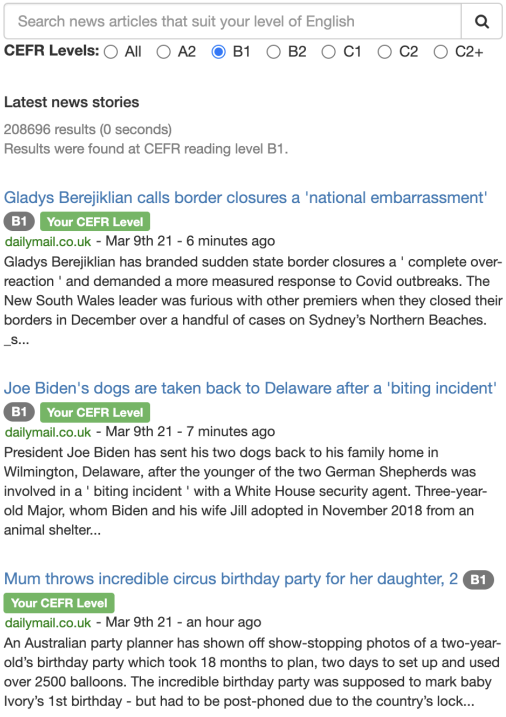
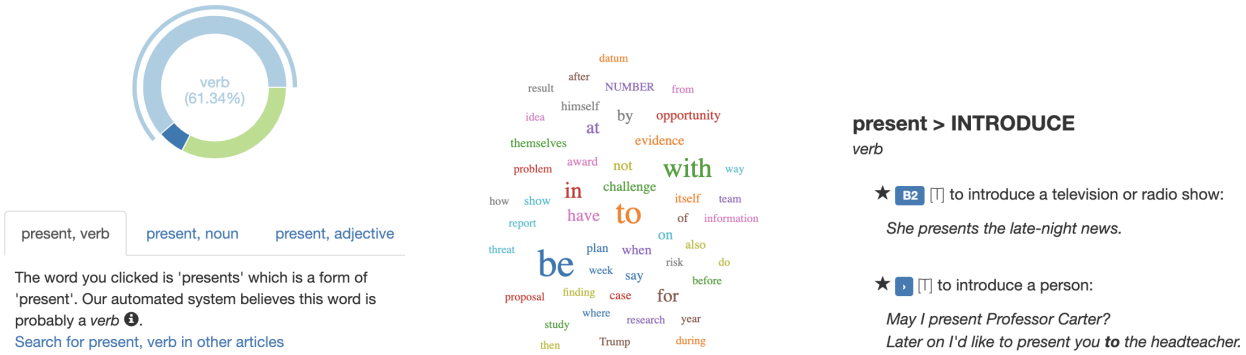


Figure 2: Screenshot: search results.

3.2 Developing one's vocabulary

Vocabulary is very important in language learning to the point that language learning itself would sometimes be equated with knowing language vocabulary [12]. To help learners

[14] `https://writeandimprove.com/`
R&I employs Write & Improve APIs developed by ELiT: `https://englishlanguageitutoring.com/`



(a) Part of speech (PoS) statistics (b) Word cloud (c) CALD word sense 'INTRODUCE'

Figure 3: Screenshots of the sections of the 'Word Information' and 'English Dictionary' panels on the UI. Here, the user clicked on the word *presents*, used as a verb. The pie chart in (a) illustrates the relative frequency of all PoS categories for the lemma *present* across all articles. The word clouds in (b) contain the 50 lemmas most frequently co-occurring with *present* as a verb in grammatical relations (where font size reflects relative frequency), and (c) shows the dictionary definition.

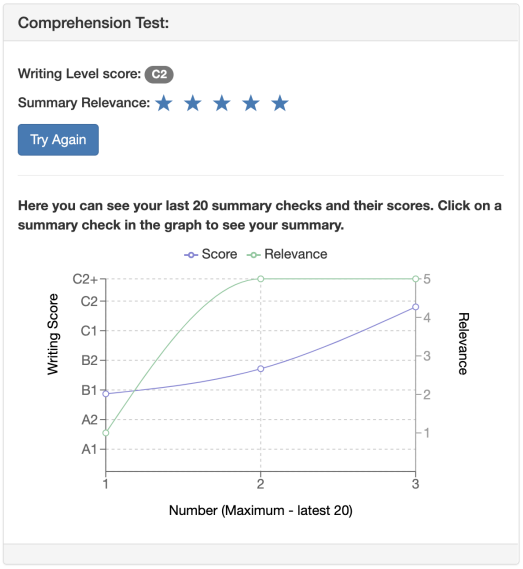


Figure 4: Screenshot: Comprehension Test panel. Learners are able to click on the graph to view previous summaries, which they can refine and re-submit.

with vocabulary acquisition and development, *R&I* allows them to select any words they do not recognise or wish to learn more about within the article view page. When a learner clicks on an unknown word, *R&I*'s UI launches two side panels for *Word Information* and *English Dictionary* (shown in Figure 3) to display information available for the word in the 'WordInfo' and 'CALD' index, respectively(§2.4).

Several searches can be performed by clicking on links within the *Word Information* panel and words within the co-occurrence word cloud. These links to search results shown in *R&I*'s search page enable learners to perform advanced, linguistically motivated searches intuitively and learn how vocabulary is used in context.

3.3 Running comprehension tests

R&I allows users to submit a summary of the article as a *comprehension test* in the *Comprehension Test* panel on the article view page (Figure 4). *R&I* automatically scores these summaries and returns a *writing score*, determined by a mature feature-based automated essay scoring (AES) model [1, 3, 20], graded on the CEFR scale via the Write & Improve API, and a *relevance score* based on the maximum sentence-level cosine similarity value, which is then converted to a score in the range 0–5 using the lexical overlap between the article and the summary [7] that shows whether the learner captured the main salient topics in the article.

3.4 Accessing reading history

All history of learner interaction with the *R&I* platform, including texts, vocabulary items and submitted summaries is available to the learners on the personal *My Reading* pages.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we presented Read & Improve, a freely available, open-access reading tutoring system that is aimed at language learners and teachers. Currently, it is a prototype system, and thence most of its components will benefit from further research on the platform. For instance, we are planning to improve our Indexing Pipeline using quality human annotated training data and user analytics that we are collecting via the *R&I* platform.

R&I records learners' actions on the UI, which in turn, will provide valuable data for use in further research and development. For example, [19] employed the comprehension test data collected by the platform to develop a new automated comprehension test (summary assessment) marking system suitable for use in *R&I*. Further, each learner's data may be useful in directly improving their learning experiences. For example, analysis of an individual learner's history could be used to tailor custom content and testing.This symbiotic relationship, developed in an ecosystem of freely available educational system benefiting from cutting-edge research, will ultimately produce a state-of-the-art ELT resource.

5. REFERENCES

[1] Ø. E. Andersen, H. Yannakoudakis, F. Barker, and T. Parish. Developing and testing a self-assessment and tutoring system. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 32–41, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[2] T. Briscoe, J. Carroll, and R. Watson. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 77–80, Sydney, Australia, July 2006. Association for Computational Linguistics.

[3] T. Briscoe, B. Medlock, and Ø. Andersen. Automated assessment of ESOL free text examinations. Technical Report UCAM-CL-TR-790, University of Cambridge, Computer Laboratory, Nov. 2010.

[4] J.-J. Chen, C.-Y. Yang, P.-C. Ho, M. C. Tsai, C.-F. Ho, K.-W. Tuan, C.-T. Tsai, W.-B. Han, and J. S. Chang. Learning to Link Grammar and Encyclopedic Information to Assist ESL Learners. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 213–218. Association for Computational Linguistics, 2019.

[5] M. Chinkina, M. Kannan, and D. Meurers. Online Information Retrieval for Language Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics—System Demonstrations*, pages 7–12. Association for Computational Linguistics, 2016.

[6] Council of Europe. Common European Framework of Reference for Languages: Learning, Teaching, Assessment, 2011.

[7] R. Cummins, H. Yannakoudakis, and T. Briscoe. Unsupervised Modeling of Topical Relevance in L2 Learner Text. In *BEA@NAACL-HLT*, 2016.

[8] Z. Dörnyei. Motivation in second and foreign language learning. *Language teaching*, 31(3):117–135, 1998.

[9] W. H. DuBay. The principles of readability. *Online Submission*, 2004.

[10] M. Heilman, L. Zhao, J. Pino, and M. Eskenazi. Retrieval of Reading Materials for Vocabulary and Reading Practice. In *Proceedings of the Third ACL Workshop on Innovative Use of NLP for Building Educational Applications*, pages 80–88. Association for Computational Linguistics, 2008.

[11] D. Hirsh and P. Nation. What vocabulary size is needed to read unsimplified texts for pleasure? *Reading in a foreign language*, 8(2):689–689, 1992.

[12] C. James. *Errors in language learning and use: Exploring error analysis*. Routledge, 2013.

[13] N. Madnani, B. B. Klebanov, A. Loukina, B. Gyawali, P. L. Lange, J. Sabatini, and M. Flor. My turn to read: An interleaved e-book reading tool for developing and struggling readers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 141–146, 2019.

[14] E. Miltsakaki and A. Troutt. Read-X: Automatic Evaluation of Reading Difficulty of Web Text. In *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, pages 7280–7286. Association for the Advancement of Computing in Education (AACE), 2007.

[15] N. Oroujlou and M. Vahedi. Motivation, attitude, and language learning. *Procedia-Social and Behavioral Sciences*, 29:994–1000, 2011.

[16] L. Rello, R. Baeza-Yates, S. Horacio, S. Bott, R. Carlini, C. Bayarri, A. Gòrriz, S. Gupta, G. Kanvinde, and V. Topac. Dyswebxia 2.0!: Accessible text for people with dyslexia (demo). In *Proceedings W4A 2013, The Paciello Group Web Accessibility Challenge*, Rio de Janeiro, Brazil, 2013.

[17] Z. Weiss, S. Dittrich, and D. Meurers. A linguistically-informed search engine to identify reading material for functional illiteracy classes. In *Proceedings of the 7th Workshop on NLP for Computer Assisted Language Learning at SLTC 2018 (NLP4CALL 2018)*, pages 79–90. Linköping Electronic Conference Proceedings 152, 2018.

[18] M. Xia, E. Kochmar, and T. Briscoe. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2016)*, pages 12–22, San Diego, California, June 2016. Association for Computational Linguistics.

[19] M. Xia, E. Kochmar, and T. Briscoe. Automatic learner summary assessment for reading comprehension. In *Proceedings of NAACL-HLT 2019*, pages 2532–2542, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[20] H. Yannakoudakis, T. Briscoe, and B. Medlock. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Generate: A NLG system for educational content creation

Saad M. Khan
FineTune Learning
saad@finetunelearning.com

Jesse Hamer
FineTune Learning
jesse@finetunelearning.com

Tiago Almeida
FineTune Learning
tiago@finetunelearning.com

ABSTRACT

We present Generate, a AI-human hybrid system to help education content creators interactively generate assessment content in an efficient and scalable manner. Our system integrates advanced natural language generation (NLG) approaches with subject matter expertise of assessment developers to efficiently generate a large number of highly customized and valid assessment items. We utilize the powerful Transformer architecture which is capable of leveraging substantive pretraining on several generic text corpora in order to produce sophisticated, context-dependent text as the basis for item creation. We present early results from experimental studies demonstrating the efficiency of our approach.

Keywords

NLP, Transformer Networks, Domain Knowledge Modeling

1. INTRODUCTION

The COVID-19 pandemic has accelerated the push towards remote delivery of formative and summative assessments and with it have arisen heightened security concerns of item pool exposure. Moreover, there is growing adoption of highly personalized and adaptive learning and assessment experiences [25] that require regularly replenished assessment item pools. These twin factors among others are placing ever growing demands on traditional processes of creating assessment content that are based in large part on manual labor, highly dependent on subject matter expertise and challenging to scale up. Furthermore, the manual generation of content and assessment items heightens the risk of incomplete, duplicate and/or redundant content. We believe advances in AI, particularly natural language generation (NLG) can help mitigate this bottleneck and open new possibilities for personalized learning experiences.

Classical natural language processing (NLP) work in this area dates back to John Wolfe's seminal work [17] that demonstrated the feasibility of automatically generating natural language questions. In recent years there has been a revival in interest, spurred in part by advances in dialogue systems such as Amazon Alexa. While traditional approaches to NLP-based assessment item generation involve a pipeline of modules such as content selection, template design and item realization [18], these have been criticized for being rigid and too reliant on arbitrary heuristic rules and having limited novelty and psychometric variability [19]. There is growing interest in developing end-to-end deep neural network based approaches that do not require customized, hand crafted rules and are better equipped to generalize across content areas [20]. A key element of such approaches is leveraging large text content databases and well annotated datasets such as BookCorpus [21], SQuAD [22] and Wikipedia. For further details on related work, readers are directed to the survey of state of the art by Kurdi et al. [26].

In this paper we present Generate, a NLG system that efficiently and in real-time creates lexically and semantically appropriate item content, dramatically speeding up assessment item authoring, freeing item writers and subject matter experts (SMEs) from unnecessary work, and can enable personalized learning and assessment experiences. At the core of Generate's content generation capabilities is the Transformer architecture [4] that leverages substantive pretraining on several generic text corpora to produce sophisticated, context-dependent text as the basis for item creation. From a small number of representative items as training samples to learn lexical and semantic structure, Generate is able to produce a wide variety of draft item content. Users utilize an intuitive graphical interface that allows selection of item stems, keys (correct answers) and distractors from a number of generated options.

In the following sections we provide technical details of our system starting with a brief review of Transformers, system implementation and architecture. Following that we present analysis from experimental studies and share thoughts on future directions.

2. TECHNICAL APPROACH AND SYSTEM DETAILS

2.1 Transformers and NLG

In order to capture the subtlety and breadth of lexical patterns necessary to faithfully generate novel assessment content, we opted to base our NLG engine on the Transformer architecture, which is capable of leveraging substantive pretraining on several generic text corpora in order to produce highly sophisticated, context-dependent token embeddings for a variety of NLP tasks. First proposed in 2017 by Vaswani et al. [4], the Transformer architecture has since revolutionized NLP research, with state-of-the-art performance on benchmarks like the broad GLUE suite of NLP tasks [5] being set by Transformer-based models such as Google's BERT [6] and OpenAI's GPT series [7, 8, 9].

The central idea of the Transformer architecture is to do away with sequential processing of text altogether, as was done traditionally with deep-learning architectures like LSTMs [10] and GRUs [11], and instead process the tokens (words, subwords, and punctuation) of text simultaneously using an operation called *attention*. The variant of attention used in the original formulation of the Transformer architecture, *scaled dot-product attention*, is defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})\ V$$

Saad Khan, Jesse Hamer and Tiago Almeida "Generate: A NLG system for educational content creation". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 736-740. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

The matrices Q and K are called the *queries* and *keys*, respectively, and each have column-dimension d, while the matrix V, called the *values*, has column-dimension d'. We consider the case when Q, K, and V are all the same matrix X, and the resulting operation is known as *self-attention*. Each row of X corresponds to a context-independent dense embedding of a single token with a small *positional encoding vector* added so that the model can take into account the position of the token in the input text. Thus, self-attention recomputes every token as a linear combination of every other token, where the weights in the linear combination depend on a scaled dot-product similarity (the $\frac{QK^T}{\sqrt{d}}$ term). In order to allow the Transformer to learn several different patterns of lexical interaction, several matrices of weights are used to compute *multi-head self-attention*:

$$\text{MultiHead}(X, X, X) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O,$$

where

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V),$$

and all of the W matrices consist of learnable weights. After multihead attention is computed, the results are aggregated and resized using a simple single-hidden-layer feedforward neural network, which has its own learnable weights. This combination of multihead attention followed by a feedforward neural network constitutes the fundamental building block of the Transformer architecture: the *Transformer block*. A *Transformer model*, then, is built by chaining together several Transformer blocks, each potentially with its own set of weights.

For its NLG engine Generate utilizes a Transformer model pre-trained for the task of next-token prediction. The Transformer architecture processes the conditioning input text in order to produce a probability distribution over all tokens in the vocabulary. We sample from the vocabulary according to this distribution, and then proceed *auto-regressively*: we process the newly sampled token and use it to produce a new probability distribution and sample a new token. We continue in this way until a maximum token limit is met, or until a stop sequence is produced (such as a newline character '\n').

2.2 How Generate Works

As illustrated in figure 1 Generate has five main system architectural components. The first is a React Javascript-based graphical user interface. Through the interface, users can select pre-uploaded AI models, generate an item, visualize and edit the item and visualize the metrics generated by the AI, allowing the user to create a complete item generation flow, from creation to validation. The user interface is linked to the second component which is the Auth0 authentication platform, a third party service specialized in secure authentication and authorization workflows. Once the user is authenticated, the GUI will connect with the third component, Hasura [2]. Hasura serves mainly as a GraphQL API to connect the GUI with the database and the serverless services. The fourth component is the item generation services (SQS Queue and Lambda Worker), which are responsible for interfacing with the NLG engine API with all the advantages of a serverless architecture [3]. The NLG engine API forms the last core component and is responsible for generating content based on the model provided.

Users begin their interaction with Generate by providing specifications of desired content including: a content map of item types and topics to be generated; user-specific writing guidelines

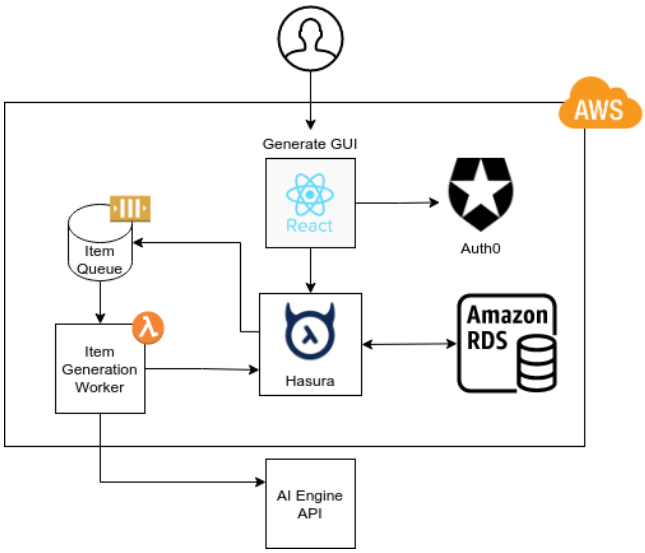


Figure 1: Generate system architecture is designed to be modular with distributed services hosted on AWS.

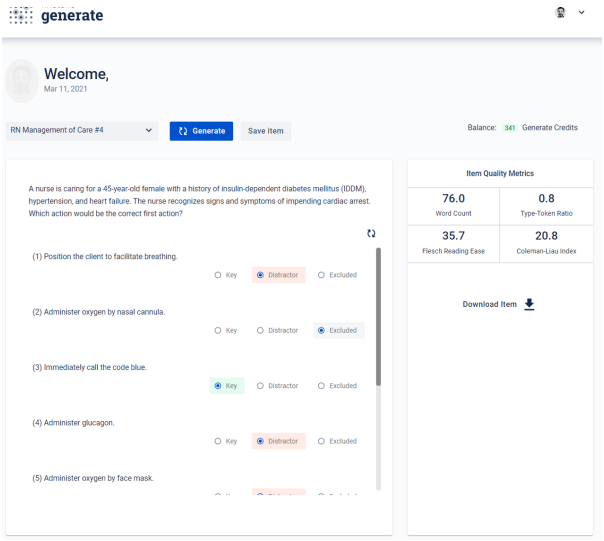


Figure 2: Generate item authoring interface. Users can select from a number of item generation models and create items on-the-fly with the click of a button.

so that domain semantics and formatting can be tailored to best practices; and specification of admissible lexical metric ranges, such as type-token ratio, Flesch Reading Ease, and the Coleman-Liau Index. These specifications constitute what is called a *project*. Along with these specifications, users must also supply a set of representative items. The number of such items is usually between 100-200 items total, although it will depend on the complexity of the content map specified in the project and the number of different types of items to be generated. At a baseline, all that is needed is the raw item, though users may supply their own item metadata to help improve the performance of our AI engine. Features such as item topic categorization, key and distractor labels, item cognitive type (recall, application, etc.), and difficulty metrics like p-value and point biserial [12] may all be used to further hone our AI models' performance.

After supplying content specifications and representative items, the *k*-means clustering algorithm [13] is applied to produce several groups of 8-10 representative items each. The clustering algorithm is predicated on a combination of user-supplied metadata, as well as numeric item features produced by the Transformer-based Universal Sentence Encoder (USE) model [14]. The goal of this clustering procedure is to produce groups of items which are semantically and stylistically homogeneous, which in turn improves the reliability of our AI engine to produce items which are coherent, semantically and factually relevant to the content domain, and stylistically appropriate according to the user's writing guidelines. Each group of representative items corresponds to a different string of conditioning input text for our AI engine, which we refer to as a *model*. Each model produces a different "flavor" of content. By building several models, we ensure that a user's content specification demands are met, and a wide diversity of items is produced while doing so.

Given a model, raw content is generated by our NLG engine and then undergoes several automated quality checks before being presented to the user. First, the raw content must pass a parse check to ensure that desired item formatting has been captured. Next, we perform an overlap check to ensure that no part of the generated content overlaps too heavily with the representative items, or with any other part of the generated content (e.g. to prevent duplicate options in a multiple choice item). For multiple choice content, users are able to specify a range of options to generate and so we also check that a sufficient number of unique options are produced. Finally, previously specified lexical metrics are computed, and we check that the generated item lies in the user-specified admissible ranges for these metrics.

As shown in figure 2, Generate offers a graphical user interface where item writers interact with our NLG engine directly to produce content. With this interface, item writers can select one of several item generation models and generate items on-the-fly with the click of a button. The item writer can then refine and annotate generated items before saving a finalized version. Generate's content generation interface allows item writers the ability to save an intermediate version of a promising item and regenerate unwanted parts. For example, with multiple choice content, one can select a key and one distractor from the list of available options, and then regenerate the remaining options to produce a fresh list to choose from. In this way, item writers can use Generate to help them rapidly ideate additional options, leading to significant speedups to the item-writing process.

Users can review generated content at any time using Generate's content dashboard. This dashboard allows users to review project-level information such as lexical metric distributions, as well as review individual items and their SME annotations. Once a selection of items has been made, users can download their content either as raw text or in QTI format.

2.3 SME Usability Experiment Results

For the content domain of nursing professional licensure, we performed two experiments with a subject matter expert (SME)/item writer in the domain. In the first experiment, the SME was asked to perform a quality review of a set of 40 items purely created by Generate NLG spanning four topic areas including biotechnology, medical assisting, nursing assisting, and practical nursing (see figure 3 for an item from this set). For a baseline of comparison, we mixed in a "calibration set" of 40 representative items produced by a separate human item writer spanning these same four topic areas. The SME was not told which items were

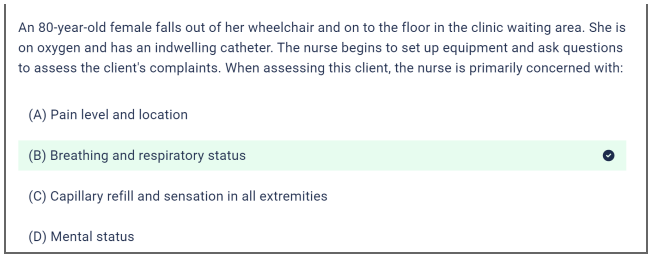


Figure 3: Sample item created by Generate. A user/SME is able to select the key/correct answer and make any edits required.

from the calibration set, and which were created by Generate. To perform the quality review, the SME was asked to check factual accuracy and topic relevance, make any necessary edits, estimate the difficulty of the item on a subjective easy/medium/hard scale, give any general comments and feedback, and assign a subjective overall quality rating on a 1-7 Likert scale (with 1 being poor and 7 being excellent). The median quality rating for Generate items was 5.5, compared to a median quality of 6 on the calibration set, with 70% of Generate items rated 5 or higher. There was also considerable overlap in the difficulty distributions, as shown in table 1. It took the SME an average of 3.75 minutes/item to perform this quality review. Compared to the SME's estimated 20-30 minutes/item to write an item manually, Generate demonstrates clear improvements to SME item writing throughput.

In the second experiment, the same SME was asked to interact directly with the Generate content generation interface to produce 50 more items in the domain of nursing professional licensure. We gave the SME five models ranging over a single nursing topic and requested that they produce ten items for each model. We captured data on generation time as well as item survival rate. For each item, the SME used Generate's content generation interface to first generate a multiple choice item with eight possible options. The SME was then asked to select the best combination of key and three distractors from these available eight options, and then perform necessary fact checking and editing. Using the Generate system, it took the SME an average of 2.7 minutes/item, including latency necessary for the system to generate the raw item. According to SME testimony, a similar exercise with a conventional item writing approach would have taken roughly 30 minutes/item, not including slowdowns due to SME fatigue and burnout. We are currently working on a number of follow-on experiments with item writers in a variety of domains including K-12 education, higher-ed and professional licensure.

Table 1: Comparison of difficulty distributions

	Easy	Medium	Hard
Generate	42.5%	40%	17.5%
Calibration	37.5%	45%	17.5%

3. DISCUSSION AND FUTURE DIRECTIONS

Our early investigations with item writers indicate a significant increase in assessment authoring throughput, which if borne out in

future and ongoing studies would mitigate bottlenecks in producing easily accessible high quality assessment items. We believe this would help enable innovations in formative assessment, personalized learning, and building customized and efficacious classroom activities.

In addition to assessment content generation, we plan to implement the following functionality to Generate over time.

3.1 Item Difficulty Estimation

Content creators must ensure that assessments adhere to a desired difficulty distribution, where we take difficulty to be measured by *p*-value (the proportion of examinees that answered a question correctly). Current methods for estimating *p*-values involve manual field-testing of provisional items, which is both time-intensive and risks item exposure, reducing the lifespan of the item. While previous work in automated difficulty estimation has employed techniques of first-order-logic [15] as well as a machine learning-based word embedding approach [16], we are exploring a blended approach which leverages structured item metadata with Transformer-based processing of unstructured item text. In this way, users can quickly recycle items which do not adhere to required difficulty specifications, thereby increasing the survival rate of items produced by Generate.

3.2 Automated Content Tagging

Tagging educational content with the most relevant learning and assessment standards such as CCSS [23], NGSS [24], etc. is one of the most critical elements in creating highly efficacious content. This enables the tracking of student skill gaps, recommendation of remediative learning resources and mastery of discipline topics, skills and cross cutting capabilities. We are currently developing a text content classification approach that can be used to delineate skills, learning objectives and core disciplinary ideas in the generated assessment items.

4. CONCLUSION

In this paper we have introduced Generate, a system that utilizes an NLG approach to significantly increase productivity of assessment content creators. Generate is built on a language modeling architecture that understands the deep semantic and lexical structure of assessment content that allow us to handle a variety of assessment domains and item types. Our system's content dashboard integrates elegantly with existing item writer workflows for item review, editing and approval. To the best of our knowledge Generate is the first NLG content authoring system designed for use in education and we believe can enable innovations in personalized learning, formative assessment and efficacious classroom activities.

5. REFERENCES

[1] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= http://doi.acm.org/10.1145/161468.16147.

[2] Hasura's open-source engine gives you instant GraphQL & REST APIs that unify your data and power modern applications - https://hasura.io/.

[3] Servereless architecture advantages - https://blog.newrelic.com/engineering/what-is-serverless-architecture/

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 5998-6008.

[5] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S. 2018a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353-355.

[6] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT (1)*, 4171-4186. DOI=10.18653/v1/N19-1423.

[7] Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. 2018. Improving language understanding by generative pre-training. *Technical report. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.*

[8] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. 2019. Language models are unsupervised multitask learners. *Technical report https://openai.com/blog/better-language-models/.*

[9] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems.*

[10] Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9, 8, 1735-1780.

[11] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Proceedings of the 2014 Conference on EMNLP* (Oct. 2014), 1724-1734. DOI=10.3115/v1/D14-1179

[12] Glass, G. and Hopkins, K. 1995. Statistical Methods in Education and Psychology (3rd ed.). *Allyn & Bacon.* ISBN 0-205-14212-5.

[13] Lloyd, S. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2, 129-137. DOI=10.1109/TIT.1982.1056489.

[14] Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C. et al. 2018. Universal sentence encoder. *Proceedings of the 2018 Conference on EMNLP: Demonstrations*, (Nov. 2018), 169-174, DOI=10.18653/v1/D18-2029.

[15] Perikos, I., Grivokostopoulou, F., Kovas, K. and Hatzilygeroudis, I. 2016. Automatic estimation of exercises' difficulty levels in a tutoring system for teaching the conversion of natural language into first-order logic. *Expert Systems* 33, 6 (Dec. 2016), 569-580. DOI=https://doi.org/10.1111/exsy.12182.

[16] Hsu, F.-Y., Lee, H.-M., Chang, T.-H. and Sung, Y.-T. 2018. Automated estimation of item difficulty for multiple-choice tests: An application of embedding techniques. *Information*

Processing & Management 54, 6 (Nov. 2018), 969-984. DOI=https://doi.org/10.1016/j.ipm.2018.06.007.

[17] Wolfe, J., 1977. Reading retention as a function of method for generating interspersed questions. *Technical report, DTIC Document*

[18] Gierl, M., Lai, H., Turner, S., 2012. Using automatic item generation to create multiple-choice test items. *Medical Education* 46, 8 (July 2012), 757-65. DOI=10.1111/j.1365-2923.2012.04289.x.

[19] Heilman, M. 2011. Automatic factual question generation from text, *Ph.D. thesis, Carnegie Mellon University.*

[20] Cervone, A., Khatri, C., Goel, R., Hedayatnia, B., Venkatesh, A., Hakkani-Tur, D. and Gabriel, R. 2019. Natural language generation at scale. *arXiv preprint arXiv:1903.08097.*

[21] Zhu. Y., Kiros, R., Zemel, R., Salakhutdinov, R., Torralba, A. and Fidler, S. 2015. Aligning books and movies. *In Proceedings of the IEEE ICCV*, 19-27.

[22] Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

[23] National Governors Association Center for Best Practices, Council of Chief State School Officers 2010. Common Core State Standards. *National Governors Association Center for Best Practices, Council of Chief State School*.

[24] NGSS Lead States. 2013. Next Generation Science Standards: For states, by states. *Washington, DC: The National Academic Press*.

[25] Pane, J. F., Steiner, E. D., Baird, M. D., & Hamilton, L. S. (2015). Continued Progress: Promising Evidence on Personalized Learning. Rand Corporation.

[26] Kurdi, G., Leo, J., Parsia, B., Sattler, U., & Al-Emari, S. (2020). A systematic review of automatic question generation for educational purposes. International Journal of Artificial Intelligence in Education, 30(1), 121-204.

Catalog: An educational content tagging system

Saad M. Khan
FineTune Learning
saad@finetunelearning.com

Joshua Rosaler
FineTune Learning
josh@finetunelearning.com

Jesse Hamer
FineTune Learning
jesse@finetunelearning.com

Tiago Almeida
FineTune Learning
tiago@finetunelearning.com

ABSTRACT

We present Catalog, an educational content classification and alignment system that tags learning and assessment content in a semantically meaningful and accurate manner. Unlike other approaches that rely on keywords or search terms and crosswalks between knowledge taxonomies, Catalog utilizes powerful NLP, specifically language models based on the Transformer architecture, to encode content in a context attentive fashion. This allows us to capture deep conceptual and contextual relations in content to classify it against a wide variety of educational standards and taxonomies. We present results from empirical studies demonstrating efficacy of our approach in classifying learning content to the Next Generation Science Standards (NGSS).

Keywords

Content tagging/classification, NLP, Transformer Networks

1. INTRODUCTION

Tagging educational content with the most relevant learning and assessment standards and education search terms is one of the most critical elements in creating highly efficacious content. This enables the tracking of student skill gaps, recommendation of remediative learning content and mastery of discipline topics, skills and cross cutting capabilities. With the ever growing volume of digital learning content and educational standards [12] the demands on tagging content are not being met by current solutions.

Current processes to tag content typically starts with raw untagged content that has to be manually reviewed, understood and analyzed by subject matter experts (SMEs) and then classified against a particular education standard e.g. the NGSS [10] resulting in the first set of foundational standards tags. Typically, these standards are hierarchical and utilize a taxonomic knowledge representation to capture the knowledge structure including core disciplinary knowledge, skills and/or cross cutting capabilities. Given the foundational tags one can transfer onto any number of desired taxonomies, for instance the Common Core State Standards [11], using taxonomy crosswalks [13]. Crosswalks are essentially mappings from one standard's taxonomy to another that have for the most part been developed by SMEs and are many times proprietary limiting their applicability.

Saad Khan, Joshua Rosaler, Jesse Hamer and Tiago Almeida "Catalog: An educational content tagging system". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 741-744. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

While in theory this process seems to offer a relatively scalable solution to the content tagging problem, in practice it is inefficient and has significant limitations. Firstly, the initial step of creating the foundational tags is manually executed and highly subjective, making it expensive and error prone. But even when that is done well the taxonomy crosswalks do not offer a perfect solution, because these crosswalks are not one-to-one mappings between the tags of one taxonomy and the other. Due to the hierarchical nature of the standards taxonomies and how they are designed and crafted by SMEs, oftentimes there are vast differences in the levels of knowledge abstraction, resulting in many-to-many mappings for the crosswalks connecting them. The end result is that for a given unit of content even when there is a foundational tag available and using an associated crosswalk, SMEs still have to make the final adjudication of the most appropriate tag in the target standard's taxonomy.

To address these challenges we have developed Catalog, an automated content classification system that leverages recent advances in NLP, specifically the Transformer architecture. This allows us to analyze educational content with richer context-aware text embeddings and pre-trained language models. We have evaluated the accuracy of our approach with promising results on an OpenStax Biology textbook [14] with ground truth NGSS tags (human experts labeled). We believe Catalog can significantly help streamline and accelerate manual workflows around content tagging and curation. These are applicable for both existing or new content, enriching existing content tags for more targeted search, discovery and recommendation as well as maintaining content alignments as educational standards evolve.

2. TECHNICAL APPROACH AND SYSTEM DETAILS

2.1 Transformers and Text Embeddings

At the core of Catalog's content classification tagging system is the Transformer architecture, first proposed by Vaswani et al in 2017 [2]. Catalog utilizes a series of pre-trained Transformer models [5, 6] to encode text-based content in vectorized features which are then further used to analyze the probability that the content is related to a textual description of the target taxonomy. Further details of this approach are presented in the following subsections. Here we present a brief overview of the Transformer architecture.

By eschewing the sequentially-processed nature of previous deep-learning NLP architectures (like LSTMs [3] and GRUs [4]) in favor of *multi-head attention*, the Transformer architecture is highly parallelizable and scalable, allowing for richer context-aware text embeddings and a substantial pre-training capacity which allows for a transfer learning approach to NLP tasks. Since its inception, research into the Transformer architecture has exploded, with variants such as Google's BERT

[5] and OpenAI's GPT series [6, 7, 8] topping several NLP benchmarks, such as the multitask GLUE suite [9].

As indicated above, Transformers are a deep-learning architecture based on the *attention mechanism*. The original formulation of the Transformer architecture used a variant known as *scaled dot-product attention*, defined as

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})\, V,$$

where the matrices Q and K are called the *queries* and *keys*, respectively, and each have column-dimension d, while the matrix V is called the *values* and has column-dimension d'. When the queries, keys, and values are all equal to some matrix X, the resulting operation is called *self-attention*. The rows of this matrix X correspond to context-independent feature vectors of the tokens of the input text, each with a small *positional encoding vector* added so that the model is aware of each token's position within the input sequence. Self-attention can be thought of as an operation which recomputes each token as a linear combination of the other tokens, where the weights of the linear combination correspond a scaled dot-product similarity score (the $\frac{QK^T}{\sqrt{d}}$ term).

In this way, (potentially long-range) interactions between tokens are captured. To allow the Transformer to learn different patterns of interaction, several matrices of learnable weights are used to compute *multi-head self-attention*:

$$\text{MultiHead}(X, X, X) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O,$$

where

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V),$$

and all of the W matrices consist of learnable weights. After multi-head self-attention is computed, the resulting feature vector is fed to a single-hidden-layer feedforward neural network for aggregation and resizing. These two consecutive operations, multi-head self-attention followed by the feedforward neural network, constitute the core of a *Transformer block*. A *Transformer model*, then, is built by chaining several Transformer blocks together, each potentially with their own set of weight matrices.

2.2 How Catalog Works

Catalog's AI-powered content tagging system utilizes a Transformer-based semantic matching engine to rank taxonomic categories by their semantic similarity to given educational content. The semantic matching algorithm works as follows. We are given a collection of textual descriptions of taxonomic categories (e.g. NGSS [10]), which we refer to as "documents," and the raw text of educational content, referred to as the "query" that needs to be classified. For each document, we produce a string of input text by combining it with the query along with a small amount of connective text. Using a Transformer model pre-trained for next token prediction, we then process the input string to convert the query tokens into feature vectors. These feature vectors are then further processed to produce probabilities for each query token, conditioned on the document text. Additionally, we process the query text by itself in order to determine unconditioned probabilities for the query tokens. Finally, a match score is produced for each document by comparing the conditioned vs. unconditioned query token probabilities and then aggregating these into a single real-valued score. Documents are then ranked according to these scores, with a higher score indicating a higher match similarity.

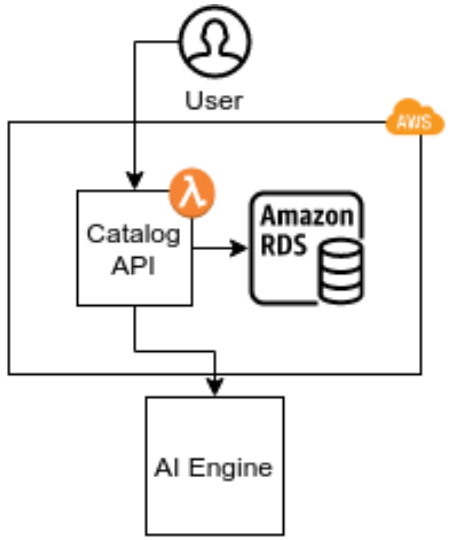
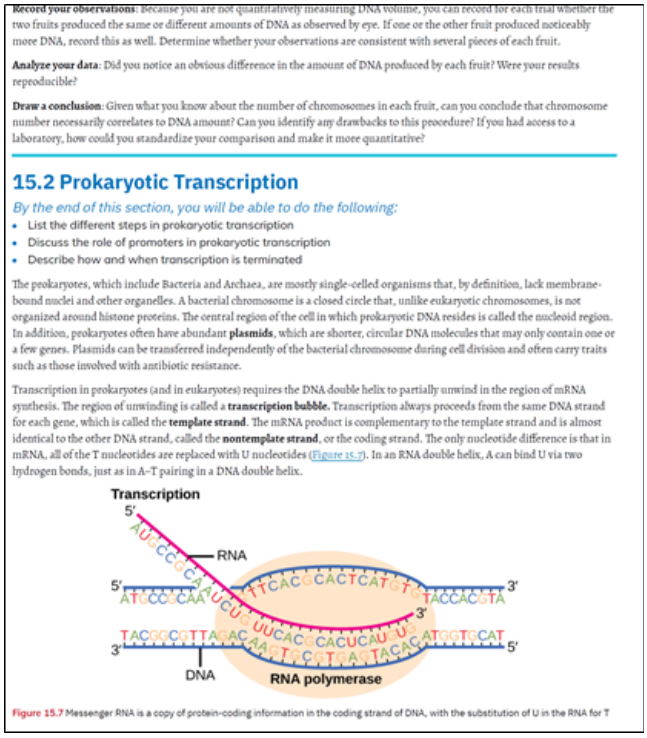


Figure 1: Catalog system architecture is designed to be modular with disturbed services hosted on AWS.


Record your observations: Because you are not quantitatively measuring DNA volume, you can record for each trial whether the two fruits produced the same or different amounts of DNA as observed by eye. If one or the other fruit produced noticeably more DNA, record this as well. Determine whether your observations are consistent with several pieces of each fruit.

Analyze your data: Did you notice an obvious difference in the amount of DNA produced by each fruit? Were your results reproducible?

Draw a conclusion: Given what you know about the number of chromosomes in each fruit, can you conclude that chromosome number necessarily correlates to DNA amount? Can you identify any drawbacks to this procedure? If you had access to a laboratory, how could you standardize your comparison and make it more quantitative?

15.2 Prokaryotic Transcription

By the end of this section, you will be able to do the following:

- List the different steps in prokaryotic transcription
- Discuss the role of promoters in prokaryotic transcription
- Describe how and when transcription is terminated

The prokaryotes, which include Bacteria and Archaea, are mostly single-celled organisms that, by definition, lack membrane-bound nuclei and other organelles. A bacterial chromosome is a closed circle that, unlike eukaryotic chromosomes, is not organized around histone proteins. The central region of the cell in which prokaryotic DNA resides is called the nucleoid region. In addition, prokaryotes often have abundant **plasmids**, which are shorter, circular DNA molecules that may only contain one or a few genes. Plasmids can be transferred independently of the bacterial chromosome during cell division and often carry traits such as those involved with antibiotic resistance.

Transcription in prokaryotes (and in eukaryotes) requires the DNA double helix to partially unwind in the region of mRNA synthesis. The region of unwinding is called a **transcription bubble.** Transcription always proceeds from the same DNA strand for each gene, which is called the **template strand**. The mRNA product is complementary to the template strand and is almost identical to the other DNA strand, called the **nontemplate strand**, or the coding strand. The only nucleotide difference is that in mRNA, all of the T nucleotides are replaced with U nucleotides (Figure 15.7). In an RNA double helix, A can bind U via two hydrogen bonds, just as in A–T pairing in a DNA double helix.

Transcription

RNA

DNA

RNA polymerase

Figure 15.7 Messenger RNA is a copy of protein-coding information in the coding strand of DNA, with the substitution of U in the RNA for T

Figure 2: Sample page from the OpenStax Biology 2e textbook used in our experiments.

System architecture and implementation wise, Catalog has two core architectural components as shown in figure 1. The first is a Lambda API Endpoint that leverages the serverless architecture and serves mainly as an interface between the user and the Transformer-based semantic query process, the "AI Engine". It authenticates users' requests, manages requests and accesses the system database. The second major component, "AI Engine"

manages content processing and returns the match scores from classification.

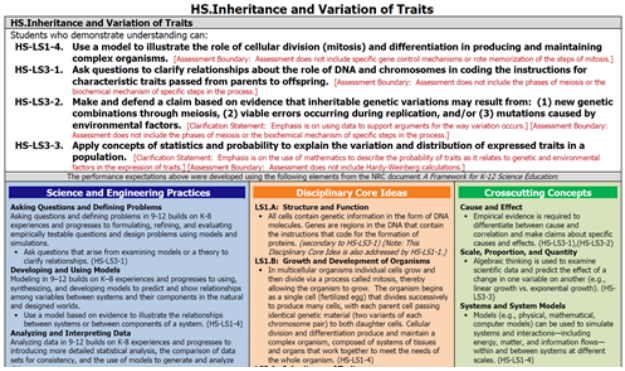


Figure 3: Sample from NGSS High School Life Science Biology performance expectation (PE) standards. Image shows 4 of the 24 unique PE standards used in our experiment.

2.3 Experimental Results

We tested the accuracy and performance of our approach on a learning content dataset extracted from the OpenStax Biology 2e high school textbook [14]. The dataset consists of approximately 500 pages of content spanning 98 chapter/subchapter sections that ranged from 410 to 545 words each. Each of the book's 98 sections is annotated with NGSS High School Life Sciences (HS-LS) performance expectation (PE) tags [10], also provided by OpenStax [14] and served as the ground truth labels in our experiment. There are a total of 24 unique Biology NGSS PE standards applicable to our dataset, essentially rendering this a 24 class classification problem. Figure 2 shows a sample from the OpenStax textbook and in figure 3 we include sample PEs from the 24 NGSS standards used in our experiment. We note that these ground truth labels are not necessarily unique: each section is associated with one to three NGSS tags.

Topic documents for the 24 PE standards were assembled from the Topic Arrangements of the NGSS that includes descriptions of PEs, Science and Engineering Practices, Disciplinary Core Ideas, Crosscutting Concepts. Because our model predicts NGSS tags for a given OpenStax section by ranking them, we assess performance by computing the *top-n overall accuracy*, that is, the proportion of predictions which have at least one ground truth label in their top-*n* ranked predictions (note that for $n = 1$, this is just the traditional overall accuracy measure). For comparison, we had an SME perform this classification exercise manually i.e. provide up to three suggested NGSS PE tags for each of the 98 book sections in our dataset. This SME is a high-school science teacher in a New York city school district and is highly experienced with the NGSS standards.

Before examining the results of this experiment, we note that one NGSS standard, HS-LS1-2, was severely overrepresented in our dataset, accounting for nearly 42% of all ground truth tags, more than 5 times the next-most-represented tag. To account for this in our accuracy computations, we decided to take 1000 random subsamples of this class, and then average the top-*n* accuracy over these subsamples. Figure 4 shows the resulting NGSS tag distribution of such a subsample.

Figure 5 shows the top-*n* accuracy averaged over the 1000 subsamples as a function of *n*. When compared to ground truth, the semantic query model achieved 51%, 73%, and 77% top-1, top-2, and top-3 overall accuracy, respectively, among the 24 NGSS PE standards. In contrast, the SME achieved 48%, 68%, and 70% top-1, top-2, and top-3 overall accuracy, respectively. It should also be noted that it took the SME 520 minutes to complete the manual classification of the dataset, whereas our system completed processing in only approximately 2 minutes.

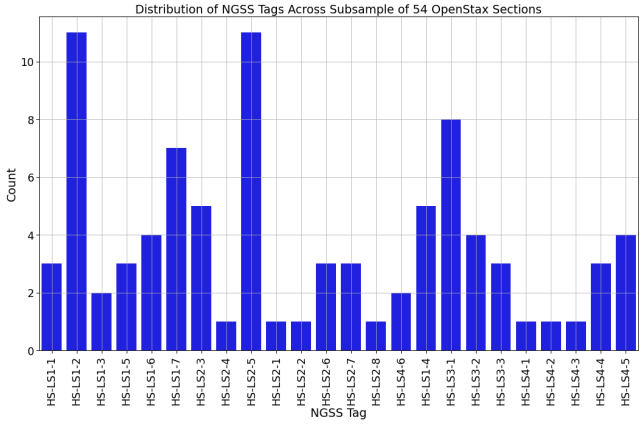


Figure 4: Distribution of NGSS tags across subsample of data. In all, 55 OpenStax sections are associated with tag HS-LS1-2, whereas each subsample randomly selects only 11 of these to be commensurate with the next most represented tag, HS-LS2-5.

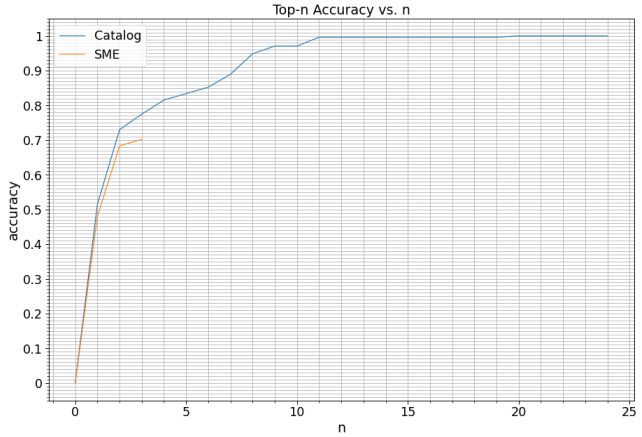


Figure 5: Top-*n* Accuracy vs. *n* for 98 items of section text from the OpenStax Biology 2e textbook, tagged against the NGSS High School Life Sciences performance expectation standards (as above, *n* is the number of top predictions within which at least one ground truth label must fall for the prediction to be counted as correct).

3. CONCLUSION

In this paper we have introduced Catalog, a NLP based content classification system that utilizes recent advances in transfer learning approaches to deeply and accurately tag educational content against popularly used learning standards. Unlike other approaches that rely on keywords or search terms and crosswalks between knowledge taxonomies, Catalog is built on a language modeling architecture that understands the deep semantic structure and relationship between concepts, topics, learning objectives and other attributes of content. We have presented early results from empirical studies demonstrating efficacy of our

approach in classifying learning content to the Next Generation Science Standards (NGSS).

4. ACKNOWLEDGEMENTS

We would like to thank Sara Vispoel for reviewing our results and insightful discussions and comments on the NGSS standards and taxonomic representations used in K-12 life science education and high school biology in particular.

5. REFERENCES

[1] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= http://doi.acm.org/10.1145/161468.16147.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 5998-6008.

[3] Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9, 8, 1735-1780.

[4] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Proceedings of the 2014 Conference on EMNLP* (Oct. 2014), 1724-1734. DOI=10.3115/v1/D14-1179

[5] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT (1)*, 4171-4186. DOI=10.18653/v1/N19-1423.

[6] Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. 2018. Improving language understanding by generative pre-training. *Technical report.* *https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf*.

[7] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. 2019. Language models are unsupervised multitask learners. *Technical report* *https://openai.com/blog/better-language-models/*.

[8] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems.*

[9] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S. 2018a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353-355.

[10] NGSS Lead States. 2013. Next Generation Science Standards: For states, by states. *Washington, DC: The National Academic Press*.

[11] National Governors Association Center for Best Practices, Council of Chief State School Officers 2010. Common Core State Standards. *National Governors Association Center for Best Practices, Council of Chief State School*.

[12] Scott-Little, C., Lesko, J., Martella, J., & Milburn, P. (2007). Early Learning Standards: Results from a National Survey to Document Trends in State-Level Policies and Practices. Early Childhood Research & Practice, 9(1), n1.

[13] Conley, D. T. (2011). Crosswalk Analysis of Deeper Learning Skills to Common Core State Standards. Educational Policy Improvement Center (NJ1).

[14] https://openstax.org/details/books/biology-2e

Are Violations of Student Privacy "Quick and Easy"? Implications of K-12 Educational Institutions' Posts on Facebook

Macy A. Burchfield[1], mburchf3@vols.utk.edu
Joshua M. Rosenberg[1], jmrosenberg@utk.edu
Conrad Borchers[2], conrad.borchers@student.uni-tuebingen.de
Tayla Thomas[1] , hqm263@vols.utk.edu
Benjamin Gibbons[3] , ben.gibbons@emory.edu
Christian Fischer[2], christian.fischer@uni-tuebingen.de

[1] University of Tennessee, Knoxville
[2] University of Tübingen
[3] Emory University

ABSTRACT

As the use of social media increases in daily life, it has also increased for institutions in the field of education. While there may be benefits for schools to use this media outlet, the privacy of students within those schools may be at risk when their names and photos are shared on such a publicly accessible domain. In this study, we analyzed the extent to which students' privacy is protected by qualitatively coding a random sample of 100 Facebook posts made by U.S. school districts from a population of over 9.3 million photo posts that we collected. Using inferential techniques, we found that students are somewhat protected compared to teachers and community members, with only 2.67% of students' detected faces able to be identified by name. The same measure for staff and community members were 4.6% and 16%, respectively. These numbers at first appear small, but if applied to the entire population, this could potentially leave between 153,218 and 1,153,844 students identifiable to anyone on the internet. We discuss the severity and scale of these privacy threats and make recommendations for research on student privacy in social media and other informal education-related contexts.

Keywords

Privacy, Social Media, Facebook, Educational Institutions, Facial Recognition

1. INTRODUCTION & PRIOR RESEARCH

As the number of people using social media has increased, the risks to the privacy of social media users have also increased [23], and this is particularly true since social media use expands into areas of our lives that it did not previously occupy. Education is one such domain in which social media use is now widespread [2, 10, 11, 13, 21, 22]—and is one domain for which the privacy risks from social media use, in general, may be compounded because of the centrality of a particularly vulnerable population, minors at school.

For students in any given school district, the use of their name or face for social media may present notable privacy concerns. As many social media posts are made publicly available, they may be accessed by unexpected sets of individuals, even by those without an account on the corresponding social networking site. Such use may pose a legitimate threat, which may be unknown to (or under-acknowledged by) teachers, administrators, and parents.

Macy Burchfield, Joshua Rosenberg, Conrad Borchers, Tayla Thomas, Benjamin Gibbons and Christian Fischer "Are Violations of Student Privacy "Quick and Easy"? Implications of K-12 Educational Institutions' Posts on Facebook". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 745-749. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

There is past research on the intersection of privacy and social media. For example, Fiesler and Proferes [6] examined what participants in social media studies thought of their data being used by others—particularly, by academic researchers. Only around one-quarter of participants in their survey study reported being comfortable with their data being used without being informed of such use.

Related lines of research explored the intersection of privacy and social media data for students. For instance, Ifenthaler and Schumacher [9] surveyed students about what they thought of their data being used in learning analytics systems. They found that while students expressed comfort with sharing some types of data (i.e., data on their course enrollments, for which less than 20% of students reported reluctance with sharing such data), for others, students were much less comfortable. Notably, highly-personal data, such as medical records, data on one's personal income, and externally-produced data, including social media, were among those that students were the least willing to share. Less than 10% of students reported being willing for externally-produced data to be used within learning analytics systems. Other scholars have shown that pre-service teachers are highly-uncomfortable with how social media companies use students' social media data, with more than two-thirds of teachers expressing discomfort with such uses [16].

While past research has explored the willingness of social media participants and students to share their data for research, a different—institutional rather than personal—context for social media use presents potentially notable privacy risks. Namely, past research has shown that both post-secondary [13] and K-12 educational institutions use social media extensively; particularly, Twitter and Facebook [10, 11]. However, to this point, no research has yet investigated privacy in the context of social media use by K-12 educational institutions.

This topic—K-12 institutions' use of social media from a privacy perspective—is relevant and timely for a number of reasons. Recent research has shown that institutions are very active on both Twitter and Facebook, being associated with more than 300,000 posts/month from the accounts of K-12 districts and schools [11]. As a consequence, there could be hundreds of thousands of students with their identities being posted in a highly-public, searchable, persistent record, and in a way that could be misused in the future. In addition, these posts may contain information that would typically be thought of as information which should not be shared publicly and widely, but which may be shared because of limited understanding of how widely such posts (on public pages) can be viewed. The audiences of institutions are likely much

greater than that of individual educators, meaning any potential privacy concerns may be much larger than that of independent users' accounts. Raising awareness of this issue may prompt some reflection on the part of those sharing this information.

This study involves an initial investigation into the extent to which students' privacy is protected through analysis of Facebook posts made by public schools and school districts. In doing so, we ask a question about the nature of Facebook. Facebook claims that its site is "quick and easy," [5] but the expediency and facility with which K-12 administrators and educators may use the platform may mean that it is also easy for school districts and schools to violate the privacy of students—with potentially difficult-to-anticipate negative ramifications at present and in the future. In particular, we aim to explore the degree to which the privacy of students might be compromised through public Facebook posts guided by the following research questions:

1. To what extent can students be identified by name and photo on public Facebook pages of schools and school districts?
2. How does the identifiability of students compare to that of staff and community members?

2. METHOD

2.1 Sample

We used a *public data mining* methodology, one that draws from *educational data mining* techniques [1, 7], but which is distinguished by the use of (largely unstructured) publicly available data, such as data from websites and social media platforms [12]. Specifically, to obtain our sample of 100 schools' and school districts' Facebook posts, we used CrowdTangle, Facebook's platform for providing academics and journalists access to data about public content on Facebook, including the content of posts and links to associated media as well as their timestamps and number of comments and likes (and other interactions) [4]. This content includes historical data from public Facebook pages with more than 50,000 likes and verified profiles. In addition, individuals with access to CrowdTangle can access public pages—but not individual users' pages.

We accessed all of the posts from K-12 institutions' public Facebook pages in the United States, having obtained the URLs to 15,728 educational institutions' Facebook pages. We did so by using the statistical software R [20] to programmatically access (or, to webscrape) their homepages using data provided by the Common Core of Data [19], and recording all links to Facebook pages from their home pages. When schools linked to the same page as the district, we considered the page as a district page. The total study population included roughly 18 million posts shared from 2005-2020, with about 9.3 million of these posts including at least one photo.

Carrying out a privacy-focused study ourselves, we took steps to protect the privacy of the individuals represented in our data. First, while we accessed and structured the data in a PostgreSQL database, we did not save the images themselves, instead using the Facebook posts and links therein to access the images through our web browser. More broadly, we determined early in our process that we were not prepared to analyze the photos algorithmically/automatically in a safe and ethical manner (e.g., using machine learning methods); we were concerned about uploading the images to a server, where they might be scanned and indexed. While we did not store the images in our database, we nevertheless took steps to protect this data, including permitting access only to authenticated members of the research team.

From the population of approximately 18 million posts, we randomly sampled 100 posts with photos for this analysis. Our random sample of posts and related coding data were stored in a private Google Sheets file stored within a University Google Account (in part because Google is less likely—based upon past legislation, lawsuits and company policies—to programmatically search the contents of educational accounts) to which only project contributors had access; this ensured that any data that could potentially be used to identify individuals was protected.

2.2 Measures

We analyzed the data qualitatively using a combination of two commonly-used qualitative analysis techniques [8], the use of priori codes that we developed based upon prior research and our research questions as well as an exploratory process that allowed us to elaborate on and to substantiate those codes and to train as coders on the use of the coding frame. In particular, we analyzed the data in two ways, as we describe next.

First, to determine *whose privacy was at risk* using our sample of 100 posts, we accessed the images from each post through photo-specific URLs that are included along with information for each post in the data. Each image was accessed and analyzed individually. When there were more than ten images included in a post, we analyzed the first ten, reasoning that these first 10 were the most likely to be seen by viewers of the post. Each post of our sample was analyzed by two trained coders to evaluate the levels of identification for all names and faces included. Upon analysis of 15 posts, we drew three categories from similar research to distinguish individuals included in posts based on their role in the school or school district community [18]:

- *Students*: Any minor assumed to be enrolled in a school and/or participating in a school hosted event or activity.
- *Staff*: Any known employee of the school or school district; including but not limited to teachers, administrators, paraprofessionals, and communications directors.
- *Community Members*: Any member of the school community who is not a verifiable student or staff member, including but not limited to parents, school board members, local business owners, and volunteers.

Second, to determine *how individuals' privacy may be threatened*, we developed a coding frame that we used to assess whether individuals' names and/or photos of individuals were shared in posts, and whether it was possible to readily connect individuals' names and photos of them. We will next describe our qualitative coding process for applying this coding frame.

2.3 Qualitative Coding

Coding proceeded by first determining the classification (student, staff, or community) of each individual detected by name or photo in a post, and then identifying the number of different first and last names included in the text of the post, as well as the number of individual faces shown in the posts' images. In particular, the following four elements were recorded for each category of individuals:

- *Number of First and Last Names in Post*

First and last names were recorded separately within each category due to the fact that staff and community members are often mentioned using their professional prefix (Mr., Mrs., Dr. , etc.) and only their last name.

- *Number of Faces in Images*

For identifying the presence of individuals' faces, a detectable face was considered to be one for which three out of four of the following features were visible without enlarging the image: 1) eyes, 2) nose, 3) ears, and 4) mouth. Any faces appearing in more than one photo within the entire post were only counted once.

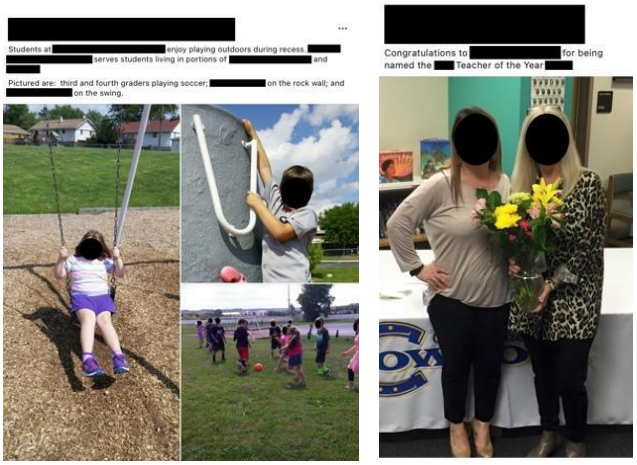
- *How many Names and Faces Connected*

We looked in posts for specific indicators of an individual's location in an image, including the order in which individuals appear in an image or labels on images. In general, identifiability criteria appeared as any text that explicitly stated which name matched with which face in which image.

Our coding included an interrater reliability check for 15 posts. Two coders coded these 15 posts individually using the coding process outlined above. Agreement percentages for detecting names, detecting faces, and identifying faces were 100%, 77.77%, and 93.33% respectively. Total agreement between coders across all codes was 92.34%.

2.4 Illustration of the Coding Process

Image 1. Example Posts



To illustrate the coding process, we provide two example posts and how we coded them above (Image 1). In the image for the first example, two student names, both first and last, are included in the text of the post, and multiple student faces are included in the three images of the post. The "third and fourth graders playing soccer" are not named individually and cannot be distinguished from each other. The two listed student names, which have been covered along with their faces for their protection, are identified by their locations in the images, thus making their faces identifiable by name as well. In the second image, the post included the name of a staff member, as well as detectable faces of two staff members. Without clarification, neither of these faces could be identified with the name mentioned.

2.5 Inferential Analysis

To analyze data to answer our first question, on how students can be identified by name and photo on public posts by K-12 educational institutions, we evaluated the percentages of student faces that were able to be identified by name; for example, if, across the 100 posts, we detected 50 student faces in images, and one was identifiable by name, then the percentage of identifiable students would be 1% (rather than 2%, because we were interested in making inferences on the basis of the number of identifiable students per post). We refer to this value in our results as the *percentage of identifiable faces per post.* Then, based on the observed frequencies (from which we calculated these percentages), we calculated binomial 95% confidence intervals for the ratio of identifiable faces and categories of faces. We did this to present an initial set of estimates for how many faces in our population of 9.3 million photo posts may be identifiable.

To answer our second research question on relative differences in identifiability of individuals from different groups, we carried out the same analysis as above (for students) for teachers and community members. Then, to compare the percentages of photos with identifiable individuals across categories, we calculated a different percentage than for RQ #1, one based not upon the number of posts (i.e., one identifiable face across 100 posts; 1%), but, rather, one based upon the total number of faces detected for people in each category. For instance, if there were 50 faces of students detected, and one was identifiable, then the percentage would be 2%; we refer to this in our results as the *percentage of identifiable faces per category sum*. This number—and comparing the confidence intervals between groups—would allow us to speak to whether individuals were differentially identifiable when photos of them were detected, even if there were, for example, far more photos of students than community members detected.

3. RESULTS

Our coding resulted in the detection (but not identification) of 299 faces in the images from the 100 posts in our sample. Of these 299 faces, only 13 (4.35% of all detected faces [2.33%, 7.32%]) were able to be identified with the individuals' name from the text of the post.

RQ #1. These 13 identifiable faces were identified within 12 individual posts from schools or districts. Student faces comprised 5 of those 12 and thus, for every 100 posts, we estimated that there were 5 identifiable student faces, representing the rate of a single identifiable student face for every twenty posts. Put another way, we estimated that 5% ([1.64%, 11.28%]) of these posts contained identifiable student faces. While this rate is relatively low, if used to make an inference about the population of photo posts we collected, this would suggest that between 153,218 and 1,053,844 students could potentially be identified via their inclusion in school or school districts' posts.

RQ #2. For students, 187 faces were detected in photos and only 5 of those 187 faces were able to be identified by their names, meaning that 2.67% ([0.87%, 6.13%]) of student faces were identifiable by name. Similar percentages are given below for each of the other categories. These numbers indicate that students and staff had a much smaller percentage of identifiable faces than that of community members. The rest of our results are shown in the table below (Table 1).

Table 1. Identifiability Percentages by Category

Category	Total # of Faces	# of ID Faces	Percentage of Identifiable Faces per Post (RQ #1)	Percentage of Identifiable Faces Per Category Sum (RQ #2)
Student	187	5	5% [1.64%, 11.28%]	2.67% [0.87%, 6.13%]

Staff	87	4	4% [1.10%, 9.92%]	4.6% [1.27%, 11.36%]
Community	25	4	4% [1.10%, 9.92%]	16% [4.54%, 36.08%]

4. DISCUSSION

4.1 Key Findings

Upon the completion of coding our sample and numerical analysis for each category, we are able to make a few important claims about the protection of student privacy. First, students comprised a majority of the faces detected in images; however, compared to the large number of student faces, less than 3% of those faces were able to be identified by their names. While this low proportion may seem to indicate that students' privacy is well-protected, the massive scope of this data (more than nine million public posts by schools or school districts) nevertheless means that many students are at-risk to be identified by both face and name by anyone with internet access if expanded to the entire data set. In short, K-12 institutions' uses of social media could introduce very widespread threats to students' privacy.

How serious are these privacy threats? An identifiable photo presents a relatively low risk compared to, for example, one's address or grade-related information being shared. However, the risk of doing so is not zero: These posts could be used to identify information about individuals, and when accessed, could potentially be used to predict their personal characteristics, even those that require making strong inferences, such as those about individuals' political identities [14]—and, potentially, other identities. Adding to the problem, we note that each of these posts not only associates a name with a photo, but also an identifiable photo to a particular location (a school or district) at a specific time. In summation, what seems like a low-risk form of identification, can reveal quite a bit of information on students, leaving their privacy vulnerable.

In addition to the number of photos of students that were able to be identified, the level of protection attached to the privacy of students was intriguing when compared to that of students and staff. More specifically, while students had the highest number of faces detected in images, their isolated level of identifiability was the lowest of all three categories. We can also note that students and staff members together have drastically lower isolated levels of identifiability compared to that of community members: Community members were generally easier to distinguish between than our other categories.

Taken together, these findings speak to concerns about privacy on social media, revealing that not only individuals' actions and posts (e.g. [6, 9, 23]), but also those of educational institutions may pose risks for the privacy of a vulnerable societal group: minors at school. They suggest that the wide use of Facebook and ease of accessing posts coupled with identifiable posts of students may make this particular use of social media a key avenue through which students' privacy is compromised. In this way, these findings add to prior research pointing out that young people may view privacy differently [17]. In addition, this research suggests to the educational data mining community that privacy risks to students may appear in unexpected contexts—and in contexts for which schools may, technically, not be violating the United States' Family Educational Rights and Privacy Act (FERPA), but which may be deserve greater scrutiny.

4.2 Limitations and Recommendations

This study represents an initial exploration of a topic that has been investigated extensively using other data sources and populations [6, 9]—and which could be investigated much further to better understand the nature of how students' privacy may be threatened due to the increasingly widespread use of social media by K-12 educational institutions. Due to the small size of our sample (compared to that of the population of photo posts), while we made some inferences from our sample to the population, these were associated with very wide ranges of plausible values: for example, we estimated that the number of identifiable students ranges from between 150,000 and more than one million, a range that makes it difficult to inform other researchers as well as administrators, educators, parents, and students about the scale of the threat to students' privacy. In addition, there are certain statistical inferences that we are unable to make at this time: For instance, with a small number of posts from varying years, we must code a larger sample to be able to model change in privacy risk over time.

It is important to consider the issue of parent consent in the context of student photos via public pages of schools and districts. While our sample data does not include specific information on each educational institution's privacy policies, there has been past research performed regarding actions such as consent forms [3]. Students' parents or legal guardians typically act as their agents of consent, which may appear to legitimize the publicizing of student faces. However, those making these crucial decisions may not have all of the necessary information to make these choices on behalf of their students.

Future research may expand on the findings presented in this study by not only coding a greater number of posts, but also coding for different features of them. For instance, we noted that because many images in the latter part of 2020 included students wearing masks, there may surprisingly be a decrease in the number of identifiable faces during the COVID-19 pandemic. Future research that aims to mitigate risks may also note some of the features of posts which protect the privacy of students, and posts by schools or districts that achieve some of the benefits of educational institutions' social media use. How accessible the posts we accessed via both the CrowdTangle [4] platform and other (authorized or unauthorized--e.g., through web-scraping) means is another topic future scholarship can explore in greater depth, as the extent to which others can reproduce our analysis has a bearing on how extensive the threats to students' privacy are. Limiting risks to students' privacy may serve as a model to inform or prompt reflection on the part of the administrators and educators using their school's or school districts' Facebook account. Finally, future research might investigate what key stakeholders--students, parents, and teachers--think of the potential privacy risks around social media use. While past research has reported that teachers are uncomfortable with how social media platforms use student data [16], our results suggest that key individuals in schools may not draw connections between this lack of comfort and how their school or district uses social media, and survey research methods may compliment our public data mining approach.

5. REFERENCES

[1] Baker, R.S., & Inventado, P.S. (2014). Educational data mining and learning analytics. In R.S. Baker & P.S. Inventatdo (Eds.), *Learning analytics* (pp. 61-75). New York: Springer

[2] Carpenter, J., Tani, T., Morrison, S., & Keane, J. (2020). Exploring the landscape of educator professional activity on Twitter: an analysis of 16 education-related Twitter hashtags. *Professional Development in Education*, 1–22. https://doi.org/10.1080/19415257.2020.1752287.

[3] Cino, D., & Vandini, C. D. (2020). "Why Does a Teacher Feel the Need to Post My Kid?": Parents and Teachers Constructing Morally Acceptable Boundaries of Children's Social Media Presence. *International journal of communication* [Online], 1153-1172. https://link.gale.com/apps/doc/A632440221/AONE?u=tel_a_utl&sid=AONE&xid=d42623382.

[4] CrowdTangle Team (2021). *CrowdTangle.* Facebook, Menlo Park, California, United States. List ID: [all-k12-institutions]. Retrieved January 15, 2021.

[5] Facebook. (n.d.). Retrieved March 12, 2021, from https://www.facebook.com

[6] Fiesler, C., & Proferes, N. (2018). "Participant" perceptions of Twitter research ethics. *Social Media+Society, 4*(1), 2056305118763366.

[7] Fischer, C., Pardos, Z. A., Baker, R. S., Williams, J. J., Smyth, P., Yu, R., ... & Warschauer, M. (2020). Mining big data in education: Affordances and challenges. *Review of Research in Education*, *44*(1), 130-160.

[8] Hatch, J. A. (2002). *Doing qualitative research in education settings.* Suny Press.

[9] Ifenthaler, D., & Schumacher, C. (2016). Student perceptions of privacy principles for learning analytics. *Educational Technology Research and Development, 64*(5), 923-938.

[10] Kimmons, R., Carpenter, J. P., Veletsianos, G., & Krutka, D. G. (2018). Mining social media divides: an analysis of K-12 US School uses of Twitter. *Learning, media and technology, 43*(3), 307-325.

[11] Kimmons, R., Rosenberg, J.M., & Allman, B. (2021). Trends in educational technology: What Facebook, Twitter, and Scopus can tell us about current research and practice. *TechTrends*, 1-12. https://link.springer.com/article/10.1007/s11528-021-00589-6

[12] Kimmons, R., & Veletsianos, G. (2018). Public internet data mining methods in instructional design, educational technology, and online learning research. *TechTrends, 62*(5), 492-500.

[13] Kimmons, R., Veletsianos, G., & Woodward, S. (2017). Institutional uses of Twitter in US higher education. *Innovative Higher Education, 42*(2), 97-111.

[14] Kosinski, M. (2021). Facial recognition technology can expose political orientation from naturalistic facial images. *Scientific Reports*, *11*(1), 1-7.

[15] Madden, M., Lenhart, A., Cortesi, S., Gasser, U., Duggan, M., Smith, A., & Beaton, M. (2013). Teens, social media, and privacy. *Pew Research Center, 21*(1055), 2-86.

[16] Marín, V. I., Carpenter, J. P., & Tur, G. (2021). Pre‐service teachers' perceptions of social media data privacy policies. *British Journal of Educational Technology, 52*(2), 519-535.

[17] Marwick, A. E., & Boyd, D. (2014). Networked privacy: How teenagers negotiate context in social media. *New Media & Society, 16*(7), 1051-1067.

[18] Michela, E., Rosenberg, J. M., Sultana, O., Burchfield, M.A., Thomas, T., & Kimmons, R. (2021, April). *"Life will eventually get back to normal": School districts' Twitter use in response to COVID-19*. Presentation at the American Educational Research Association Annual Meeting, Orlando, FL.

[19] National Center for Education Statistics. (2021). *Common core of data.* https://nces.ed.gov/ccd/

[20] R Core Team (2021). *R: A language and environment for statistical computing*. https://www.r-project.org/s

[21] Romero-Hall, E., Kimmons, R., & Veletsianos, G. (2018). Social media use by instructional design departments. *Australasian Journal of Educational Technology, 34*(5).

[22] Rosenberg, J. M., Greenhalgh, S. P., Koehler, M. J., Hamilton, E. R., & Akcaoglu, M. (2016). An investigation of state educational Twitter hashtags (SETHs) as affinity spaces. *E-learning and Digital Media, 13*(1-2), 24-44.

[23] Smith, M., Szongott, C., Henne, B., & Von Voigt, G. (2012, June). Big data privacy issues in public social media. In *2012 6th IEEE international conference on digital ecosystems and technologies (DEST)* (pp. 1-6). IEEE.

Towards Explainable Student Group Collaboration Assessment Models Using Temporal Representations of Individual Student Roles

Anirudh Som
Center for Vision Technologies
SRI International
anirudh.som@sri.com

Sujeong Kim
Center for Vision Technologies
SRI International
sujeong.kim@sri.com

Bladimir Lopez-Prado
Center for Education
Research and Innovation
SRI International
bladimir.lopez-prado@sri.com

Svati Dhamija
Center for Vision Technologies
SRI International
svati.dhamija@sri.com

Nonye Alozie
Center for Education
Research and Innovation
SRI International
maggie.alozie@sri.com

Amir Tamrakar
Center for Vision Technologies
SRI International
amir.tamrakar@sri.com

ABSTRACT

Collaboration is identified as a required and necessary skill for students to be successful in the fields of Science, Technology, Engineering and Mathematics (STEM). However, due to growing student population and limited teaching staff it is difficult for teachers to provide constructive feedback and instill collaborative skills using instructional methods. Development of simple and easily explainable machine-learning-based automated systems can help address this problem. Improving upon our previous work, in this paper we propose using simple temporal-CNN deep-learning models to assess student group collaboration that take in temporal representations of individual student roles as input. We check the applicability of dynamically changing feature representations for student group collaboration assessment and how they impact the overall performance. We also use Grad-CAM visualizations to better understand and interpret the important temporal indices that led to the deep-learning model's decision.

Keywords

K-12, Education, Collaboration Assessment, Explainable, Deep-Learning, CNN, Grad-CAM, Cross-modal Analysis.

1. INTRODUCTION

Collaboration is considered a crucial skill, that needs to be inculcated in students early on for them to excel in STEM fields [24, 6]. Traditional instruction-based methods [14, 7] can often make it difficult for teachers to observe several student groups and identify specific behavioral cues that contribute or detract from the collaboration effort [20, 15, 25]. This has resulted in a surge in interest to develop machine-learning-based automated systems to assess student group collaboration [17, 11, 12, 8, 1, 9, 26, 23, 21, 4, 27, 22].

In our earlier work we developed a multi-level, multi-modal conceptual model that serves as an assessment tool for individual student behavior and group-level collaboration quality [2, 3]. Using the conceptual model as a reference, in a different paper we developed simple MLP deep-learning models that predict student group collaboration quality from histogram representations of individual student roles [22]. Please refer to the following papers for more information and for the illustration of the conceptual model [2, 3, 22]. Despite their simplicity and effectiveness, the MLP models and histogram representations lack explainability and insight into the important student dynamics. To address this, in this paper we focus on using simple temporal-CNN deep learning models to check the scope of dynamically changing temporal representations for student group collaboration assessment. We also use Grad-CAM visualizations to help identify important temporal instances of the task performed and how they contribute towards the model's decision.

Paper Outline: Section 2 provides necessary background on the different loss functions used, dataset description and the temporal features extracted. Section 3 describes the experiments and results. Section 4 concludes the paper.

2. BACKGROUND

2.1 Cross-Entropy Loss Functions

The categorical-cross-entropy loss is the most commonly used loss function to train deep-learning models. For a classification problem with C classes, let us denote the input variables as $\mathbf{x}$, ground-truth label vector as $\mathbf{y}$ and the predicted probability distribution as $\mathbf{p}$. Given a training sample $(\mathbf{x}, \mathbf{y})$, the categorical-cross-entropy (CE) loss is defined as

Anirudh Som, Sujeong Kim, Bladimir Lopez-Prado, Svati Dhamija, Nonye Alozie and Amir Tamrakar "Towards Explainable Student Group Collaboration Assessment Models Using Temporal Representations of Individual Student Roles". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 750-754. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Table 1: Coding rubric for Level A and Level B2.

Level A	Level B2
Effective [E]	Group guide/Coordinator [GG]
Satisfactory [S]	Contributor (Active) [C]
Progressing [P]	Follower [F]
Needs Improvement [NI]	Conflict Resolver [CR]
Working Independently [WI]	Conflict Instigator/Disagreeable [CI]
	Off-task/Disinterested [OT]
	Lone Solver [LS]

Table 2: Inter-rater reliability (IRR) measurements.

Level	Average Agreement	Cohen's Kappa
A	0.7046	0.4908
B2	0.6741	0.5459

$$\text{CE}_{\mathbf{x}}(\mathbf{p}, \mathbf{y}) = -\sum_{i=1}^{C} \mathbf{y}_i \log(\mathbf{p}_i) \tag{1}$$

Here, $\mathbf{p}_i$ denotes the predicted probability of the i-th class. Note, both $\mathbf{y}$ and $\mathbf{p}$ are of length C, with $\sum_i \mathbf{y}_i = \sum_i \mathbf{p}_i = 1$. From Equation 1, it's clear that for imbalanced datasets the learnt weights of the model will be biased towards classes with the most number of samples in the training set. Additionally, if the label space exhibits an ordered structure, the categorical-cross-entropy loss will only focus on the predicted probability of the ground-truth class while ignoring how far off the incorrectly predicted sample actually is. These limitations can be addressed to some extent by using the ordinal-cross-entropy (OCE) loss function [22], defined in Equation 2.

$$\begin{aligned} \text{OCE}_{\mathbf{x}}(\mathbf{p}, \mathbf{y}) &= -(1+w)\sum_{i=1}^{C} \mathbf{y}_i \log(\mathbf{p}_i) \\ w &= |\text{argmax}(\mathbf{y}) - \text{argmax}(\mathbf{p})| \end{aligned} \tag{2}$$

Here, $(1+w)$ represents the weighting variable, argmax returns the index of the maximum valued element and $|.|$ returns the absolute value. When training the model, $w = 0$ for correctly classified training samples, with the ordinal-cross-entropy loss behaving exactly like the categorical-cross-entropy loss. However, for misclassified samples the ordinal-cross-entropy loss will return a higher loss value. The increase in loss is proportional to how far away a sample is misclassified from its ground-truth class label.

2.2 Dataset Description

We collected audio and video recordings from 15 student groups, across five middle schools. Out of the 15 groups, 13 groups had 4 students, 1 group had 3 students, and 1 group had 5 students. The student volunteers completed a brief survey that collected their demographic information and other details, e.g., languages spoken, ethnicity and comfort levels with science concepts. Each group was tasked with completing 12 open-ended life science and physical science tasks, which required them to construct models of different science phenomena as a team. They were given one hour to complete as many tasks possible, which resulted in 15 hours of audio and video recordings. They were provided logistic and organization instructions but received no help in group dynamics, group organization, or task completion.

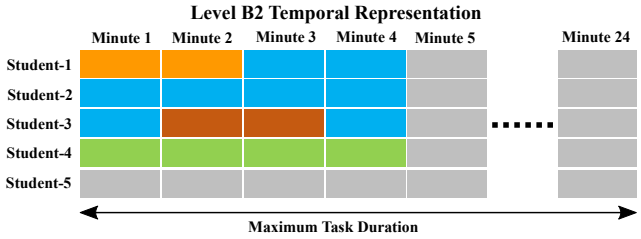


Figure 1: Level B2 temporal representation for a group having only 4 students and finishing the assigned task in 4 minutes. Colored cells illustrate the different Level B2 codes as described in Table 1, and the gray cells represent empty or unassigned codes.

Next, the data recordings were manually annotated by education researchers at SRI International. For the rest of the paper we will refer to them as coders/annotators. In our hierarchical conceptual model [2, 3], we refer to the collaboration quality annotations as Level A and individual student role annotations as Level B2. The coding rubric for these two levels is described in Table 1. Both levels were coded by three annotators. They had access to both audio and video recordings and used ELAN (an open-source annotation software) to annotate. A total of 117 tasks were coded by each annotator, with the duration of each task ranging from 5 to 24 minutes. Moderate-agreement was observed across the coders as seen from the inter-rater reliability measurements in Table 2.

Level A codes represent the target label categories for our classification problem. To determine the ground-truth Level A code, the majority vote (code) across the three annotators was used as the ground-truth. For cases where a majority was not possible, we used the Level A code ordering depicted in Table 1 to determine the median as ground-truth of the three codes. For example, if the three coders assigned *Satisfactory, Progressing, Needs Improvement* for the same task then *Progressing* would be used as the ground-truth label. Note, we did not observe a majority Level A code for only 2 tasks. To train the machine learning models we only had 351 data samples (117 tasks $\times$ 3 coders).

2.3 Temporal Representation

In our dataset, the longest task was little less than 24 minutes, due to which the length for all tasks was also set to 24 minutes. Level B2 was coded using fixed-length 1 minute segments, as illustrated in Figure 1. Due to its fixed-length nature, we assigned an integer value to each B2 code, i.e., the seven B2 codes were assigned values from 1 to 7. The value 0 was used to represent segments that were not assigned a code. For example, in Figure 1 we see a group of 4 students completing a task in just 4 minutes, represented by the colored cells. The remaining 20 minutes and the 5^{th} student is assigned a value zero, represented by the gray cells. Thus for each task, Level B2 temporal features will have a shape 24×5, with 24 representing number of minutes and 5

representing number of students in the group.

Baseline Histogram Representation: We compare the performance of the temporal representations against simple histogram representations [22]. The histogram representations were created by pooling over all the codes observed over the duration of the task and across all the students. Note, only one histogram was generated per task, per group. Once the histogram is generated we normalize it by dividing by the total number of codes in the histogram. Normalizing the histogram removes the temporal aspect of the task. For example, if group-1 took 10 minutes to solve a task and group-2 took 30 minutes to solve the same task, but both groups were assigned the same Level A code despite group-1 finishing the task sooner. The raw histogram representations of both these groups would look different due to the difference in number of segments coded. However, normalized histograms would make them more comparable. Despite the normalized histogram representation being simple and effective, it fails to offer any insight or explainability.

3. EXPERIMENTS

Network Architecture: For the temporal-CNN deep learning model we used the temporal ResNet architecture described in [28]. The ResNet architecture uses skip connections between each residual block to help avoid the vanishing gradient problem. It has shown state-of-the-art performance in several computer vision applications [10]. Following [28], our ResNet model consists of three residual blocks stacked over one another, followed by a global-average-pooling layer and a softmax layer. The number of filters for each residual block was set to 64, 128, 128 respectively. The number of learnable parameters for the B2 temporal representations is 506949. We compare the performance of the ResNet model to the MLP models described in our previous work. Interested readers should refer to [22] for more information about the baseline MLP model that was used with the histogram representation.

Training and Evaluation Protocol: All models were developed using Keras with TensorFlow backend [5]. We used the Adam optimizer [13] and trained all models for 500 epochs. The batch-size was set to one-tenth of the number of training samples during any given training-test split. We optimized over the Patience and Minimum-Learning-Rate hyperparameters, that were set during the training process. We focused on these as they significantly influenced the model's classification performance. The learning-rate was reduced by a factor of 0.5 if the loss did not change after a certain number of epochs, indicated by the Patience hyperparameter. We saved the best model that gave us the lowest test-loss for each training-test split. We used a round-robin leave-one-group-out cross validation protocol. This means that for our dataset consisting of g student groups, for each training-test split we used data from $g-1$ groups for training and the left-out group was used as the test set. This was repeated for all g groups and the average result was reported. For our experiments $g = 14$ though we have temporal representations from 15 student groups. This is because all samples corresponding to the *Effective* class were found only in one group. Due to this reason and because of our cross-validation protocol we do not see any test samples for the *Effective* class.

Table 3: Weighted precision, weighted recall and weighted F1-score Mean±Std for the best MLP and ResNet models under different settings.

Feature	Classifier	Weighted Precision	Weighted Recall	Weighted F1-Score
B2 Histogram	SVM	84.45±13.43	73.19±16.65	76.92±15.39
	MLP - Cross-Entropy Loss	83.72±16.50	86.42±10.44	84.40±13.85
	MLP - Cross-Entropy Loss + Class-Balancing	83.93±17.89	85.29±14.37	84.16±16.23
	MLP - Ordinal-Cross-Entropy Loss	86.96±14.56	88.78±10.36	**87.03±13.16**
	MLP - Ordinal-Cross-Entropy Loss + Class-Balancing	86.73±14.43	88.20±9.66	86.60±12.54
B2 Temporal	ResNet - Cross-Entropy Loss	84.75±13.21	83.10±11.92	82.72±12.74
	ResNet - Cross-Entropy Loss + Class-Balancing	84.03±15.13	83.28±11.42	82.97±12.84
	ResNet - Ordinal-Cross-Entropy Loss	85.24±15.68	87.23±10.52	85.56±13.38
	ResNet - Ordinal-Cross-Entropy Loss + Class-Balancing	84.34±15.75	87.88±11.22	**85.68±13.58**

3.1 Temporal vs Histogram Representations

Here, we compare the performance of the ResNet and MLP models. Using the weighted F1-score performance, Table 3 summarizes the best performing ResNet and MLP models for the different feature-classifier variations. The table also provides the weighted precision and recall metrics. Bold values in the table represent the best classifier across the different feature-classifier settings. The ordinal-cross-entropy loss with or without class-balancing shows the highest weighted F1-score performance for both feature types. Here, class-balancing refers to weighting each data sample by a weight that is inversely proportional to the number of data samples corresponding to that sample's ground-truth label.

At first glance, the ResNet models perform slightly less than the MLP models. This could easily lead us to believe that simple histogram representations are enough to achieve a higher classification performance than the corresponding temporal representations. However, despite the performance differences, the temporal features and ResNet models help better explain and pin-point regions in the input feature space that contribute the most towards the model's decision. This is important if one wants to understand which student roles are most influential in the model's prediction. We will go over this aspect in more detail in the next section.

3.2 Grad-CAM Visualization

Grad-CAM uses class-specific gradient information, flowing into the final convolutional layer to produce a coarse localization map that highlights the important regions in the input feature space [19]. It is primarily used as a post hoc analysis tool and is not used in any way to train the model. Figure 2 illustrates how Grad-CAM can be used for our classification problem. We show two different samples from the Satisfactory, Progressing and Needs Improvement classes respectively. Each sample shows a group consisting of 4 students that completed the task in 5 to 8 minutes. Technically one can obtain C Grad-CAM maps for a C-class classification problem. Here, the samples shown correspond to the class predicted by the ResNet model, which is also the ground-truth class. It's clear how the Grad-CAM highlights regions in the input feature space that contributed towards the correct prediction. For instance, in the Needs Improvement examples, the Grad-CAM map shows the highest weight on the fourth minute. At that time for the first example, the codes for three of the students become Off-task/Disinterested. Similarly, for the second example we notice three of the students become Lone Solvers and the

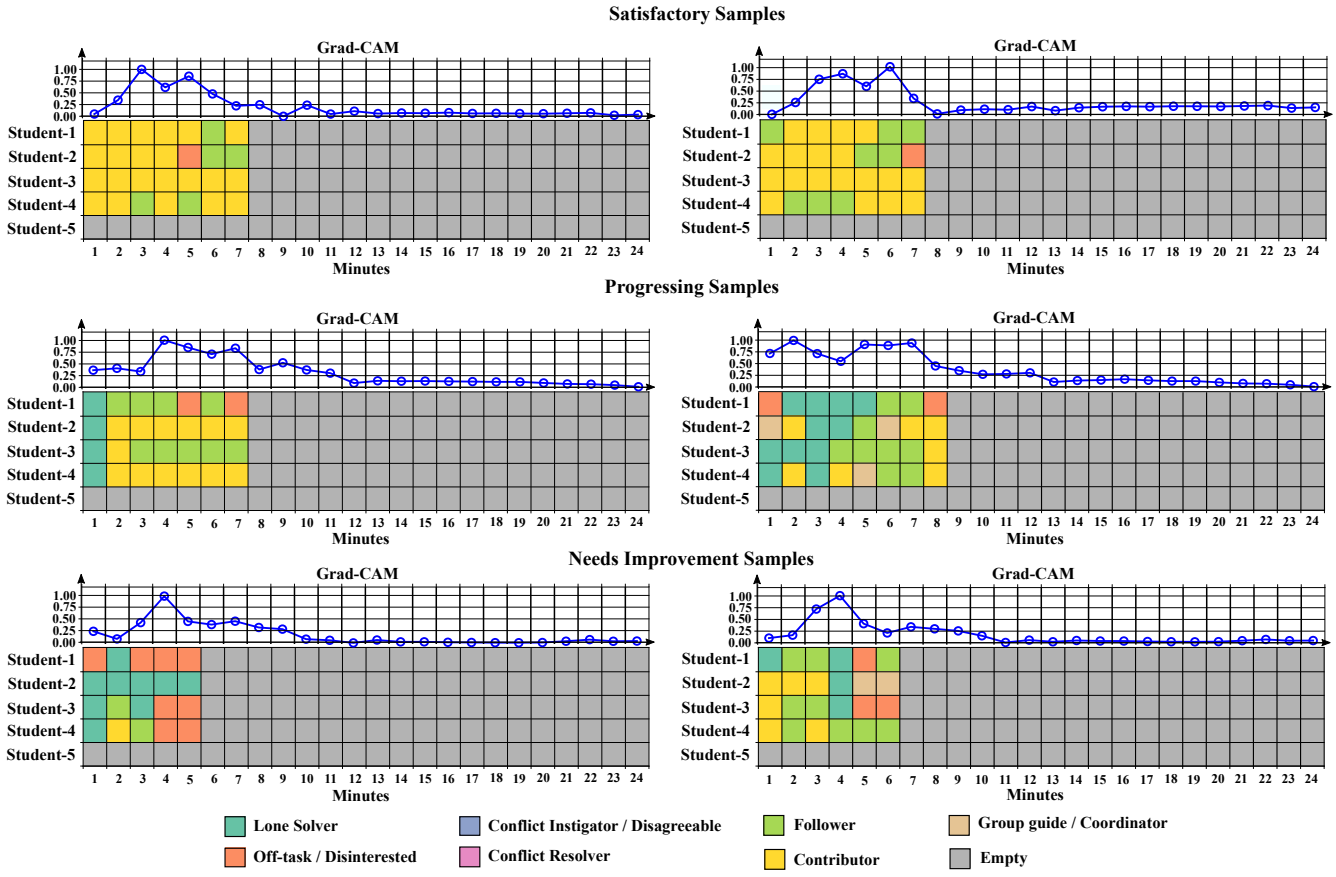


Figure 2: Grad-CAM visualization for two different temporal samples from different Level-A classes.

fourth student becomes a Follower. This is in stark contrast to the minute before when two of the students were Followers and the other two were Contributors. We also notice less importance being given to the Empty codes. These changes in roles and the Grad-CAM weights across the task make sense and help promote explainability in our deep learning models.

4. CONCLUSION

In this paper we proposed using simple temporal representations of individual student roles together with temporal ResNet deep-learning architectures for student group collaboration assessment. Our objective was to develop more explainable systems that allow one to understand which instances in the input feature space led to the deep-learning model's decision. We suggested use of Grad-CAM visualization along the temporal dimension to assist in locating important time instances in the task performed. We compared the performance of the proposed temporal representations against simpler histogram representations from our previous work [22]. While histogram representations can help achieve high classification performance, they do not offer the same key insights that one can get using the temporal representations.

Limitations and Future Work: The visualization tools and findings discussed in this paper can help guide and shape future work in this area. Having said that our approach can be further extended and improved in several ways. For example, we only discuss Grad-CAM maps along the temporal dimensions. This only allows us to identify important temporal instances of the task but does not focus on the important student interactions. The current setup does not tell us which subset of students are interacting and how that could affect the overall group dynamic and collaboration quality. To address this we intend on exploring other custom deep-learning architectures and feature representation spaces. We also plan on using other tools like LIME [18] and SHAP [16]. These packages compute the importance of the different input features and help towards better model explainability and interpretability. Also we only focused on mapping deep learning models from individual student roles to overall group collaboration. In the future we intend on exploring other branches in the conceptual model, described in [2, 3]. We also plan on developing recommendation systems that can assist and guide students to improve themselves by suggesting what they need to take on. The same system could also be tweaked specifically for teachers to give them insight on how different student interactions could be improved to facilitate better group collaboration.

5. ACKNOWLEDGEMENT

This work was supported in part by NSF grant number 2016849.

6. REFERENCES

[1] G. Alexandron, J. A. Ruipérez-Valiente, and D. E. Pritchard. Towards a general purpose anomaly detection method to identify cheaters in massive open online courses. 2020.

[2] N. Alozie, S. Dhamija, E. McBride, and A. Tamrakar. Automated collaboration assessment using behavioral analytics. 2020.

[3] N. Alozie, E. McBride, and S. Dhamija. Collaboration conceptual model to inform the development of machine learning models using behavioral analytics. 2020.

[4] A. R. Anaya and J. G. Boticario. Application of machine learning techniques to analyse student interactions and improve the collaboration process. *Expert Systems with Applications*, 38(2):1171–1181, 2011.

[5] F. Chollet et al. Keras. https://keras.io, 2015.

[6] W. R. Daggett and D. S. GendroO. Common core state standards initiative. *International center*, 2010.

[7] N. Davidson and C. H. Major. Boundary crossings: Cooperative learning, collaborative learning, and problem-based learning. *Journal on excellence in college teaching*, 25, 2014.

[8] C. Genolini and B. Falissard. Kml: A package to cluster longitudinal data. *Computer methods and programs in biomedicine*, 104(3):e112–e121, 2011.

[9] Z. Guo and R. Barmaki. Collaboration analysis using object detection. In *EDM*, 2019.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] K. Huang, T. Bryant, and B. Schneider. Identifying collaborative learning states using unsupervised machine learning on eye-tracking, physiological and motion sensor data. *International Educational Data Mining Society*, 2019.

[12] J. Kang, D. An, L. Yan, and M. Liu. Collaborative problem-solving process in a science serious game: Exploring group action similarity trajectory. *International Educational Data Mining Society*, 2019.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] J. S. Krajcik and P. C. Blumenfeld. *Project-based learning*. na, 2006.

[15] M. L. Loughry, M. W. Ohland, and D. DeWayne Moore. Development of a theory-based assessment of team member effectiveness. *Educational and psychological measurement*, 67(3):505–524, 2007.

[16] S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[17] J. M. Reilly and B. Schneider. Predicting the quality of collaborative problem solving through linguistic analysis of discourse. *International Educational Data Mining Society*, 2019.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[19] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[20] K. A. Smith-Jentsch, J. A. Cannon-Bowers, S. I. Tannenbaum, and E. Salas. Guided team self-correction: Impacts on team mental models, processes, and effectiveness. *Small Group Research*, 39(3):303–327, 2008.

[21] A. Soller, J. Wiebe, and A. Lesgold. A machine learning approach to assessing knowledge sharing during collaborative learning activities. 2002.

[22] A. Som, S. Kim, B. Lopez-Prado, S. Dhamija, N. Alozie, and A. Tamrakar. A machine learning approach to assess student group collaboration using individual level behavioral cues. In *European Conference on Computer Vision Workshops*, pages 79–94. Springer, 2020.

[23] D. Spikol, E. Ruffaldi, and M. Cukurova. Using multimodal learning analytics to identify aspects of collaboration in project-based learning. Philadelphia, PA: International Society of the Learning Sciences., 2017.

[24] N. L. States. *Next generation science standards: For states, by states*. The National Academies Press, 2013.

[25] S. Taggar and T. C. Brown. Problem-solving team behaviors: Development and validation of bos and a hierarchical factor structure. *Small Group Research*, 32(6):698–726, 2001.

[26] L. Talavera and E. Gaudioso. Mining student data to characterize similar behavior groups in unstructured collaboration spaces. In *Workshop on artificial intelligence in CSCL. 16th European conference on artificial intelligence*, pages 17–23. Citeseer, 2004.

[27] H. Vrzakova, M. J. Amon, A. Stewart, N. D. Duran, and S. K. D'Mello. Focused or stuck together: Multimodal patterns reveal triads' performance in collaborative problem solving. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 295–304, 2020.

[28] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

The CommonLit Ease of Readability (CLEAR) Corpus

Scott Crossley
Georgia State University
Atlanta, GA 30303
scrossley@gsu.edu

Aron Heintz
CommonLit
Washington, DC 20003
aron.heintz@commonlit.org

Joon Choi
Georgia State University
Atlanta, GA 30303
jchoi92@gsu.edu

Jordan Bachelor
Georgia State University
Atlanta, GA 30303
jbatchelor@gsu.edu

Mehrnoush Karimi
Georgia State University
Atlanta, GA 30303
mkarimi3@student.gsu.edu

Agnes Malatinszky
CommonLit
Washington, DC 20003
agnes.malatinszky@commonlit.org

ABSTRACT

In this paper, we introduce the Commonlit Ease of Readability (CLEAR) corpus. The corpus provides researchers within the educational data mining community with a resource from which to develop and test readability metrics and to model text readability. The CLEAR corpus improves on previous readability corpora include size (N = ~5,000 reading excerpts), the breadth of the excerpts available, which cover over 250 years of writing in two different genres, and the readability criterion used (teachers' ratings of text difficulty for their students). This paper discusses the development of the corpus and presents reliability metrics as well as initial analyses of readability.

Keywords

Text readability, corpus linguistics, pairwise comparisons

1. INTRODUCTION

Reading is an essential skill for academic success. One important way to support and scaffold literacy challenges faced by students is to match text difficulty to their reading abilities. Providing students with texts that are accessible and well matched to their abilities helps to ensure that students better understand the text and, over time, can help readers improve their reading skills. Readability formulas, which provide an overview of text difficulty, have shown promise in more accurately benchmarking students with their text difficulty level, allowing students to read texts at target readability levels.

Most educational texts are matched to readers using traditional readability formulas like Flesch-Kincaid Grade Level (FKGL) [19] or commercially available formulas such as Lexile [30] or the Advantage-TASA Open Standard (ATOS) [29]. However, both types of readability formulas are problematic. Traditional readability formulas lack construct and theoretical validity because they are based on weak proxies of word decoding (i.e., characters or syllables per word) and syntactic complexity (i.e., number or words per sentence) and ignore many text features that are important components of reading models including text cohesion and semantics. Additionally, many traditional readability formulas were normed using readers from specific age groups on small corpora of texts taken from specific domains. Commercially available readability formulas are not publicly available, may not have rigorous reliability tests, and may be cost-prohibitive for many schools and districts let alone teachers.

In this paper, we introduce the open-source the CommonLit Ease of Readability (CLEAR) corpus. The corpus is a collaboration between CommonLit, a non-profit education technology organization focused on improving reading, writing, communication, and problem-solving skills, and Georgia State University (GSU) with the end goal of promoting the development of more advanced and open-source readability formulas that government, state, and local agencies can use in testing, materials selection, material creation, and other applications commonly reserved for readability formulas. The formulas that will be derived from the CLEAR corpus will be open-source and ostensibly based on more advanced natural language processing (NLP) features that better reflect the reading process. The accessibility of these formulas and their reliability should lead to immediate uptake by students, teachers, parents, researchers, and others, increasing opportunities for meaningful and deliberate reading experiences. We outline the importance of text readability along with concerns about previous readability formulas below. As well, we present the methods used to develop the CLEAR corpus. We then examine how well traditional and newer readability formulas correlate with the reading criteria reported in the CLEAR corpus and discuss next steps.

2. TEXT READABILITY

Text readability can be defined as the ease with which a text can be read (i.e., processed) and understood in terms of the linguistic features found in that text [9][27]. However, in practice, many readability formulas are more focused on measuring text understanding (e.g., [18]) than text processing.

Text comprehension is generally associated with word sophistication, syntactic complexity, and discourse structures [17][31], three features whose textual elements relate to text complexity. For example, many studies have revealed that word sophistication features such as sound and spelling relationships between words [16][25], word familiarity and frequency [15], and word imageability and concreteness [28] can result in faster word processing and more accurate word decoding. The meaning of

Scott Crossley, Aron Heintz, Joon Choi, Jordan Batchelor, Mehrnoush Karimi and Agnes Malatinszky "The CommonLit Ease of Readability (CLEAR) Corpus". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 755-760. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

words, or semanticity, also plays an important role in text readability, in that readers must be able to recognize words and know their meaning [26]. Therefore, word semanticity and larger text segments can facilitate the linking of common themes and easier processing based on background knowledge and text familiarity [1][23].

Effective readers should also be able to parse syntactic structures within a text to help organize main ideas and assign thematic roles where necessary [13][26]. Two features that allow for quicker syntactic parsing are words or morphemes per t-unit [8] and sentence length [21]. Parsing information in the text helps readers develop larger discourse structures that result in a discourse thread [14]. These structures, which relate to text cohesion, can be partially constructed using linguistic features that link words and concepts within and across syntactic structures [12]. Sensitivity to these cohesion structures allows readers to build relationships between words, sentences, and paragraphs, aiding in the construction of knowledge representations [4][20][23]. Moreover, such sensitivity can help readers understand larger discourse segments in texts [11][26].

Traditional readability formulas tend use only proxy estimates for measuring lexical and syntactic features. Moreover, they disregard the semantic features and discourse structures of texts. For instance, these formulas ignore text features including text cohesion [4][20][23][24] and style, vocabulary, and grammar, which play important roles in text readability [1]. Additionally, the reading criteria used to develop traditional formulas are often based on multiple-choice questions and cloze tests, two methods that may not measure text comprehension accurately [22]. Finally, traditional readability formulas are suspect because they have been normed using readers from specific age groups and using small corpora of texts from specific domains.

Newer formulas, both commercial and academic, generally outperform traditional readability formulas. These formulas rely on more advanced NLP features, although this may not be the case with commercial formulas for which text features within the formulas are proprietary and, thus, not publicly available. Newer formulas come with their own issues though. For instance, commercially available formulas, such as the Lexile framework [30] and the Advantage-TASA Open Standard for Readability (ATOS) formula [29], often lack suitable validation studies. In addition, accessing commercially available formulas may come at a financial cost that is unaffordable for some schools and education technology organizations. Academic formulas such as the Crowdsourced Algorithm of Reading Comprehension (CAREC) [7] have been validated through rigorous empirical studies, are transparent in their underlying features, and are free to the public. However, the datasets on which they have been developed, while much larger than traditional readability formulas, can still be considered as relatively small and specific. The populations the formulas are trained on (i.e., adults) may also not generalize well to other target populations like young students.

3. CURRENT STUDY

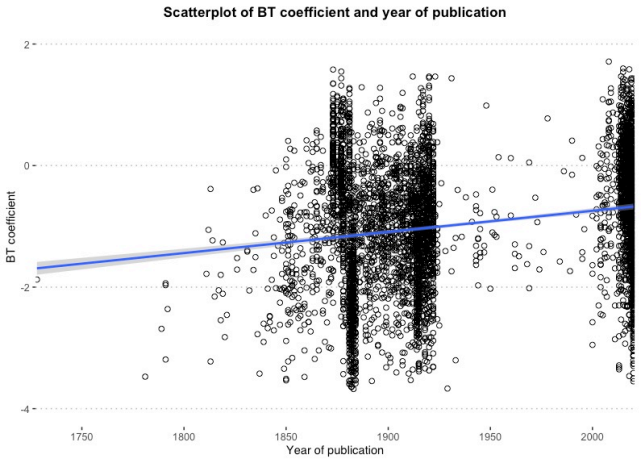
We hope to spur innovation to address many of the concerns noted above in reference to both traditional and newer readability formulas by publicly releasing the CommonLit Ease of Readability (CLEAR) corpus as well as hosting an open-source competition to develop readability formulas based on the CLEAR corpus. We hope that these formulas outperform existing readability formulas and can be used to better match 3rd-12th grade students to texts, thus improving learning outcomes in primary and secondary classrooms.

4. THE CLEAR CORPUS

4.1 Corpus Collection

We collected text excerpts from the CommonLit organization's database, Project Gutenberg, Wikipedia, and dozens of other open digital libraries. Excerpts were selected from the beginning, middle, and end of texts and only one sample was selected per text. Text excerpts were selected to be between 140-200 words, with all excerpts beginning and ending at an idea unit (i.e., we did not cut excerpts in the middle of sentences or ideas). The text excerpts were written between 1791 and 2020, with the majority of excerpts selected between 1875 and 1922 (when copyrights expired) and between 2000 and 2020 (when non-copyright texts were available on the internet). Visualizations of these trends are available in Figure 1.

Figure 1



Excerpts were selected from two genres: informational and literature texts. We started with an initial sample of ~7,600 texts. Each excerpt was read by at least two raters and judged on acceptability. The two major criteria for acceptability were the likelihood of being used in a 3rd-12th grade classroom and whether or not the topic was appropriate. We used Motion Picture Association of America (MPAA) ratings (e.g., G, PG, PG-13) to flag texts by appropriateness. Texts that were flagged as potentially inappropriate were then read by an expert rater and either included or excluded from the corpus. We also conducted automated searches for traumatic terms (e.g., terms related to racism, genocide, or sexual assault). Any excerpt flagged for traumatic terms was also reviewed by an expert rater. Lastly, we limited author representation such that each author had no more than 12 excerpts within the corpus. After removing excerpts based on these criteria, we were left with 4793 excerpts. These excerpts were copy-edited to ensure texts did not contain grammatical, syntactic, and spelling errors. Punctuation was also standardized in the texts, as were line-breaks. Lastly, selected archaic spellings (e.g., to-day, Servia) were replaced with modern spellings (e.g., today, Serbia) and identified British English spellings were converted to American spellings.

4.2 Human Ratings of Readability

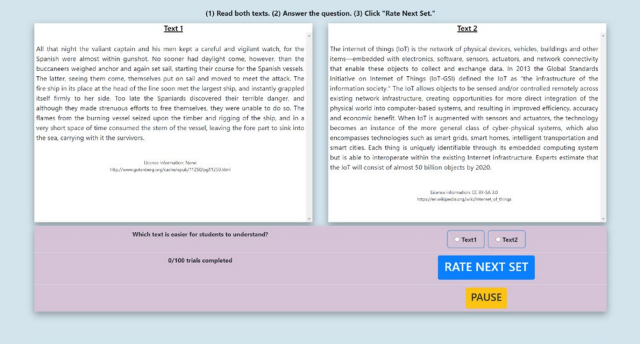
We recruited ~1,800 teachers from the CommonLit teacher pool through an e-mail marketing campaign. Teachers were asked to participate in an online collection experiment. They were

expected to read 100 pairs of excerpts and make a judgment for each pair as to which excerpt was easier to understand. Teachers were paid $50 in an Amazon gift card for their participation.

4.3 Data Collection Site

We developed an online data collection website. The basic format of the site was to show two excerpts side by side and ask participants to judge which of the two texts would be easier for a student to understand using a checkbox format. There were two additional buttons on the website. The first moved the participant to the next comparison and the second allowed participants to pause the experiment. The website also included a progress tally to show participants how many comparisons they had made (see Figure 2 for screenshot of pairwise comparison task).

Figure 2



The website first provided participants with informed consent and an overview of the expectations. The website then collected simple demographic information and survey information about reading/writing and television habits. Participants were then given a practice excerpt comparison to familiarize them with the design. After the practice comparison, participants moved forward with the data collection. Excerpts were paired randomly, and excerpts were shown on either the right or left-side panel randomly. The licensing information and the uniform resource locator (URL) for each text were displayed on the bottom side of each panel. Participants were redirected to a break screen after completing every 20 comparisons. The break screen showed how much time (in total and per comparison) the participant had spent on the task. A button allowing the participant to continue to the next comparison appeared after spending one minute on the break screen, meaning that the participants were required to take at least a one-minute break per 20 comparisons. After completing 100 comparisons, the participants were given a completion code that they could redeem for the gift card. The website was written in Python, JavaScript, CSS, and HTML. The website was housed on a cloud server.

4.4 Participant Reliability

Of the ~1,800 participants that initially logged into the experiment, 1,198 completed the entire experiment. However, not all participant data was kept. We removed participants who did not complete the entire experiment. We also removed participants to increase the reliability of the pairwise scores based on deviant patterns and time spent on judgments. In terms of deviant patterns, we removed all participants who selected excerpts in either the right or left panel more than 70% of the time. We also removed participants who had binary patterns of selecting left/right or right/left panels more than 20 times in a row. In terms of time spent on judgments, we removed participants who spent less than 10 seconds on average per comparison and/or spent a median time under 5 seconds. After removing participants based on patterns and time, we were left with data from 1,116 participants. Those participants made 111,347 overall comparison judgments (M = 99.773 judgments per participant). On average, each excerpt was read 46.47 times and participants spent an average of 101.36 seconds per judgment. However, we did not remove participants for taking too long on judgments, especially since pauses were allowed. Thus, our data for time was right skewed.

4.5 Pairwise Rankings for Readability

To calculate pairwise comparison scores for the human judgments of text ease, we used a Bradley-Terry model [3]. A Bradley-Terry model describes the probabilities of the possible outcomes when items are judged against one another in pairs (see Equation 1). The Bradley-Terry model ranks documents by difficulty based on each excerpt's probability to be easier than other excerpts. The model creates a maximum likelihood estimate which iteratively converges towards a unique maximum that defines the ranking of the excerpts (i.e., the easiest texts have the highest probability).

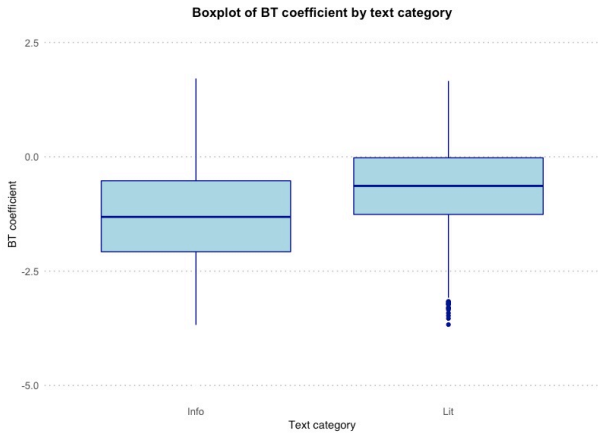
Equation 1: Bradley-Terry Model

$$P(\llbracket text \rrbracket\_i \text{ more difficult than } \llbracket text \rrbracket\_j) = \gamma\_i/(\gamma\_i+\gamma\_j)$$

After computation, the Bradley-Terry model provides a coefficient for each text along with a standard error. We examined both coefficients and standard errors for outliers. We found 52 texts that had a coefficient with a standard deviation greater than 2.5 and additional 17 excerpts with a standard error greater than 0.65. These were removed from the final dataset leaving us with a sample size of 4,724. We conducted two additional analyses of the final data set in terms of differences in Bradley-Terry coefficients between informational and literature texts and trends in the coefficients as a function of time of publication for the texts.

As expected, we found significant differences between informational and literature texts such that informational texts were rated significantly more difficult ($t(4723) = -20.95$, $p < .001$), with a moderate effect size ($d = -0.61$). See Figure 2 for a box plot depicting this difference in text categories. In addition, we used a Pearson's correlation test to test whether Bradley-Terry coefficients were correlated with the texts' year of publication, finding a weak correlation, $r(4722) = .20$, $p < .001$). Thus, more recent passages were often rated as simpler than older passages (see Figure 1).

Figure 3



4.6 Pairwise Scoring Validation Checks

To examine convergent validity for the pairwise scores, we examined correlations between the scores and classic and newer readability formulas. The formulas we included were Flesch Reading Ease, Flesch Kincaid Grade Level, the New Dale-Chall, and the Crowdsourced Algorithm of Reading Comprehension [7]. All formulas were calculated using the Automatic Readability Tool for English (ARTE) [6]. ARTE provides free and easy access to a wide range of readability formulas and is available at linguisticanalysistools.org. ARTE automatically calculates different readability formulas for batches of texts (i.e., thousands of texts can be run at a time) and produces readability scores for individual texts in an accessible spreadsheet output. ARTE was developed to help educators and researchers easily process texts and derive different readability metrics allowing them to compare that output and choose formulas that best fit their purpose. The tool is written in Python and is packaged in a user-friendly GUI that is available for use in Windows and Mac operating systems. Correlations for this analysis are reported in Table 1.

Table 1: Correlations between readability formulas and text ease

	FRE	FKGL	NDC	CAREC
Text ease	0.547	-0.517	-0.557	-0.582
FRE		-0.913	-0.829	-0.726
FKGL			0.676	0.579
NDC				0.739

*FRE = Flesch Reading Ease, FKGL = Flesch Kincaide Grade Level, NDC = New Dale Chale, CAREC = Crowdsourced Algorithm of Reading Comprehension

The results indicate strong overlap between the four selected readability formulas and the text ease scores reported by the Bradley-Terry model. The strongest correlations were reported for CAREC while the weakest correlations were reported for FKGL. While strong, the correlations indicate that the readability formulas only predict around 27%-34% of the variance in the reading ease scores. Thus, there are opportunities for improvement in future readability formulas.

5. DISCUSSION

In this paper, we introduced the CommonLit Ease of Readability (CLEAR) corpus. The corpus provides researchers within the educational data mining community with a resource from which to develop and test readability metrics and to model text readability. The CLEAR corpus has a number of improvements over previous readability corpora, which are discussed below.

First, the CLEAR corpus is much larger than any available corpora that provide readability criterion based on human judgments. While there are large corpora that provide leveled texts (e.g., The Newsela corpus), these corpora only provide indications of reading ability based on levels of simplification (i.e., beginning texts as compared to intermediate texts). The corpora do not provide readability criterion for individual texts. Individual reading criteria, like that reported in the CLEAR corpus, allows for the development of linear models of text readability. While there are other corpora that have reading criteria for individual texts, the corpora are much smaller (N = ~20 - 600 texts), and they do not contain the breadth of texts found in the CLEAR corpus. The size of the CLEAR corpus ensures wide sampling and variance such that readability formulas derived from the corpus should be strongly generalizable to new excerpts.

The breadth of excerpts found in the CLEAR corpus is an additional strength. The corpus was curated from the excerpts available on the CommonLit website, all of which have been specially leveled for a particular grade level. The CommonLit texts were supplemented by hand selected excerpts taken from Project Gutenberg, Wikipedia, and dozens of other open digital libraries. The text excerpts were published over a wide range of years (1791-2020) and are representative of two genres commonly found in the K-12 classroom: informational and literary genres. The texts were read by experts to ensure they matched excerpts used in the K-12 classroom and checked for appropriateness using MPAA ratings. All texts were hand edited, so that grammatical, syntactic, and spelling errors were limited, while punctuation was minimally standardized to honor the authors' expression and style.

A final strength is the reading criteria developed for the CLEAR Corpus. Previous studies have developed reading criteria based on cloze tests or multiple-choice tests, both of which may not measure text comprehension accurately [22]. Additionally, while many readability formulas are marketed for K-12 students, their readability criteria are based on a different population of readers. The best example of this is Flesch-Kincaid Grade Level, which was developed using reading tests administered to adult sailors. We bypass these concerns, to a degree, by collecting judgments from schoolteachers about how difficult the excerpts would be for their students to read. This provides greater face validity for our readability criteria, which should translate into greater predictive power for readability formulas developed on the CLEAR corpus.

Lastly, while the purpose of the CLEAR corpus is for the development of readability formulas, the corpus includes meta-data that will allow for interesting and important sub-analyses. These analyses would include investigations into readability differences based on year of publication, genre, author, and standard errors, among many others. The sub-analyses afforded by the CLEAR corpus will allow greater understandings of how variables beyond just the language features in the excerpts influence text readability.

6. FUTURE DIRECTIONS

The next step for the CLEAR corpus is an online data science competition to promote the development of new open-science readability formulas. The competition will be hosted within an online community of data scientists and machine learning engineers who will enter a competition to develop readability formulas using only the reading excerpts and the reported standard errors to predict the Bradley-Terry ease of reading co-efficient scores. Prize money will be offered to increase the likelihood of participation. Once winners from the competition are announced, the winning readability formulas will be included in ARTE so that access to the formulas is readily available to teachers, students, administrators, and researchers. ARTE will also be expanded to include an online interface and a functional API. The online interface will allow end-users to easily upload texts to analyze for readability to better match texts to readers. The API will allow other educational technologies to include text readability formulas in their systems to help select texts for online students.

7. ACKNOWLEDGMENTS

We want to thank Kumar Garg and Schmidt Futures for their advice and support for making this work possible. We also thank other researchers who helped develop the CLEAR corpus and to

the nearly two thousand teacher participants who provided judgments of text readability.

8. REFERENCES

[1] Bailin, A. & Grafstein, A. (2001). The linguistic assumptions underlying readability formulae: A critique. Language & Communication, 21(3), 285-301. DOI = https://doi.org/10.1016/S0271-5309(01)00005-2.

[2] Benjamin, R. G. (2012). Reconstructing readability: Recent developments and recommendations in the analysis of text difficulty. Educational Psychology Review, 24(1), 63-88. DOI = https://doi.org/10.1007/s10648-011-9181-8.

[3] Bradley, R.A. & Terry, M.E. (1952). Rank analysis of incomplete block designs. I. The method of paired comparisons. Biometrika, 39, 324-345. DOI = https://doi.org/10.2307/2334029.

[4] Britton, B.K. & Gülgöz, S. (1991). Using Kintsch's computational model to improve instructional text: Effects of repairing inference calls on recall and cognitive structures. Journal of Educational Psychology, 83, 329-345.

[5] Chall, J. S., & Dale, E. (1995). Readability revisited: The new Dale-Chall readability formula. Brookline Books.

[6] Choi, J.S., & Crossley, S.A. (2020) *Machine Readability Applications in Education*. Paper presented at Advances and Opportunities: Machine Learning for Education (NeurIPS 2020).

[7] Crossley, S. A., Skalicky, S., & Dascalu, M. (2019). Moving beyond classic readability formulas: New methods and new models. Journal of Research in Reading, 42(3-4), 541-561. DOI = https://doi.org/10.1111/1467-9817.12283.

[8] Cunningham, J.W., Spadorcia, S.A., Erickson, K.A., Koppenhaver, D.A., Sturm, J.M., & Yoder, D.E. (2005). Investigating the instructional supportiveness of leveled texts. Reading Research Quarterly, 40(4), 410-427. DOI = https://doi.org/10.1598/RRQ.40.4.2.

[9] Dale, E., & Chall, J. S. (1948). A formula for predicting readability: Instructions. Educational research bulletin, 37-54.

[10] Flesch, R. (1948). A new readability yardstick. Journal of Applied Psychology, 32(3), 221-233. DOI = https://doi.org/10.1037/h0057532.

[11] Gernsbacher, M.A. (1990). Language comprehension as structure building. Hillsdale, NJ: Erlbaum.

[12] Givón, T. (1995). Functionalism and grammar. Philadelphia: John Benjamins.

[13] Graesser, A.C., Swamer, S.S., Baggett, W.B. & Sell, M.A. (1996). New models of deep comprehension. In B.K. Britton & A.C. Graesser (Eds.), Models of understanding text, (pp. 1-32). Mahwah, NJ: Erlbaum.

[14] Grimes, J.E. (1975). The thread of discourse. The Hague, Netherlands: Mouton

[15] Howes, D.H. & Solomon, R.L. (1951). Visual duration thresholds as a function of word probability. Journal of Experimental Psychology, 41(6), 401-410. DOI = https://doi.org/10.1037/h0056020.

[16] Juel, C. & Solso, R.L. (1981). The role of orthographic redundancy, versatility and spelling-sound correspondences in word identification. (1981). In M.L. Kamil (Ed.), Directions in reading: Research and instruction, (pp. 74-82). Rochester, NY: National Reading Conference.

[17] Just, M. A., & Carpenter, P. A. (1980). A theory of reading: From eye fixations to comprehension. Psychological Review, 87, 329-354. DOI = https://doi.org/10.1037/0033-295x.87.4.329.

[18] Kate, R.J., Luo, X., Patwardhan, S., Franz, M., Florian, R., Mooney, R.J. et al. (2010, August). Learning to predict readability using diverse linguistic features. In Proceedings of the 23rd International Conference on Computational Linguistics, (pp. 546-554). USA: Association for Computational Linguistics

[19] Kincaid JP, Fishburne RP Jr, Rogers RL, Chissom BS. Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy enlisted personnel. Research Branch Report. Millington, TN: Naval Technical Training Command, 1975: 8-75. DOI = https://doi.org/10.21236/ada006655.

[20] Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. Psychological Review, 95, 163-182. DOI = https://doi.org/10.1037/0033-295X.95.2.163.

[21] Klare, G.R. (1984). Readability. In P.D. Pearson, R. Barr, M.L. Kamil, & P. Mosenthal (Eds.), Handbook of reading research (Vol. 1, pp. 681-744). New York: Longman.

[22] Magliano, J.P., Millis, K., Ozuru, Y. & McNamara, D.S. (2007). A multidimensional framework to evaluate reading assessment tools. In D.S. McNamara (Ed.), Reading comprehension strategies: Theories, interventions, and technologies, (pp. 107-136). Mahwah, NJ: Lawrence Erlbaum Associates Publishers.

[23] McNamara, D.S. & Kintsch, W. (1996). Learning from texts: Effects of prior knowledge and text coherence. Discourse Processes, 22, 247-288. DOI = https://doi.org/10.1080/01638539609544975.

[24] McNamara, D. S., Kintsch, E., Butler-Songer, N., & Kintsch, W. (1996). Are good texts always better? Interactions of text coherence, background knowledge, and levels of understanding in learning from text. Cognition and Instruction, 14, 1-43. DOI = https://doi.org/10.1207/s1532690xci1401_1.

[25] Mesmer, H.A. (2005). Decodable text and the first grade reader. Reading & Writing Quarterly, 21(1), 61-86. DOI= https://doi.org/10.1080/10573560590523667.

[26] Mesmer, H.A., Cunningham, J.W., & Hiebert, E.H. (2012). Toward a theoretical model of text complexity for the early grades: Learning from the past, anticipating the future. Reading Research Quarterly, 47(3), 235-258. DOI = https://doi.org/10.1002/rrq.019

[27] Richards, J. C., Platt, J., & Platt, H. (1992). Longman Dictionary of Language Teaching and Applied Linguistics. London: Longman.

[28] Richardson, J.T.E. (1975). The effect of word imageability in acquired dyslexia. Neuropsychologia, 13(3), 281-288. DOI = https://doi.org/10.1016/0028-3932(75)90004-4.

[29] School Renaissance Inst., Inc. (2000). The ATOS[TM] readability formula for books and how it compares to other formulas. Madison, WI: School Renaissance Inst., Inc.

[30] Smith, D., Stenner, A.J., Horabin, I., Smith, M. (1989). The Lexile scale in theory and practice: Final report. Washington, DC: MetaMetrics.

[31] Snow, C. (Ed.) (2002). Reading for understanding: Toward an R & D program in reading comprehension. Santa Monica, CA: Rand.

Predictive Sequential Pattern Mining via Interpretable Convolutional Neural Networks

Lan Jiang
University of Illinois Urbana–Champaign
Champaign, IL, USA
lanj3@illinois.edu

Nigel Bosch
University of Illinois Urbana–Champaign
Champaign, IL, USA
pnb@illinois.edu

ABSTRACT

We present an algorithm using interpretable convolutional neural networks for mining sequential patterns from event log data. The key to our approach is utilizing structured regularization to achieve sparse parameter values that closely resemble the results of typical pattern mining algorithms, and allows the learned convolution filters to be interpreted easily. Our method can handle both sequences of individual, unique elements and concurrent multiple-element sequences, which represents most situations where sequences may occur in logs of student actions. We applied our structured regularization method to a self-supervised problem predicting future actions from past actions in two different educational datasets as example applications. Furthermore, we generated features from the learned patterns to evaluate the utility of patterns and trained a supervised model with these features to predict academic outcomes via transfer learning. Our algorithm improves the correlation of sequences with outcomes by an average of $r = .131$ on one dataset and $r = .101$ on the other dataset versus a traditional sequential pattern mining algorithm. Finally, we visualize the extracted patterns and demonstrate that they can be interpreted as a sequence of actions.

Keywords

Interpretability, pattern mining, convolutional neural networks, sequential data

1. INTRODUCTION

Convolutional neural networks (CNNs) have been successfully applied to various applications in educational data mining [2, 9, 15, 16, 22, 24]. However, CNNs have a major shortcoming in terms of transparency, because they typically form "end to end" models that make high-level inferences from low-level inputs through a series of opaque layers. Thus, resulting models are often hard to understand and interpret. As a consequence, both instructors and students do not know what kinds of student behaviors actually impact predictions, which is important for understanding and supporting students' learning behaviors and instructors' regulation of student learning. Our work addresses this interpretability issue with CNNs to provide useful features for student modeling applications that utilize CNNs.

One particular application that requires understanding the patterns learned by a model (or any method) is sequential pattern mining. The typical approaches for sequential pattern mining is to identify a set of elements that frequently co-occur; in sequence data, that corresponds to finding a set of events, items, locations, etc. that often happen sequentially in the data [1, 3, 5, 6, 7, 8, 13, 14, 23, 25]. However, those methods of pattern discovery suffer from problems like pattern explosion [21], which occurs when the number of frequent patterns is myriad and the importance (or usefulness) of patterns is uncertain. Consequently, the large number of patterns and prioritization of common patterns can be especially problematic in educational data where low-support patterns may be of interest (e.g., when examining uncommon patterns specific to students from underrepresented demographic groups [20]). In addition, existing methods do not consider the context of a pattern. We aim to train convolutional neural networks that have inherently interpretable features (i.e., discrete absence/presence of a specific student event, like watching a video or posting to a discussion forum) and enforce learning of patterns that predict context. We can thus interpret and utilize these patterns in downstream tasks in the same way that patterns mined via sequence mining methods are.

This paper aims to train a CNN with self-supervised learning to produce a model that can predict future actions based on sequences of events, thereby encouraging the model to learn *predictive* sequences (rather than frequent, unique, or other criteria). The features learned by self-supervised neural networks can be shared by various downstream tasks, as has been demonstrated in previous research. In this paper, we report results from a CNN trained to predict future student activities from past activities, and which thus captures event dependencies.

To show the effectiveness of the proposed method, we applied it two large datasets of student actions logged in learning environments, and utilized transfer learning to predict student outcomes with features derived from the discovered patterns. Specifically, after learning the patterns from student data, we generated feature representations from each

Lan Jiang and Nigel Bosch "Predictive Sequential Pattern Mining via Interpretable Convolutional Neural Networks". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 761-766. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

pattern and trained a supervised model to predict students' grade outcomes. We demonstrate that in several cases our results outperformed and were more stable than a typical sequential pattern mining method.

In summary, our contributions include two parts:

1. We trained interpretable CNN filters to explicitly learn patterns consisting of either mutually-exclusive (unique) or concurrent (co-occurring) elements (i.e., actions).

2. We evaluated the quality of patterns learned with our method in a transfer learning task involving prediction of students' outcomes in two datasets.

2. APPROACH

Our goal is to find frequent, predictive patterns with fixed length given sequences of actions done by students (or events, or items). Each step can contain either a single unique action or multiple concurrent actions (depending on the dataset), which can be regarded as sequential actions, events, activities, or other categorical values. In this section, we first explain the framework of the unique event pattern detector, which is the simplest case and perhaps the most widely-applicable. We then describe the solution for the multiple concurrent events pattern detector, as an illustration of how the unique-element approach can be generalized to other variations of the problem. We also describe a "warm-up" strategy, which is necessary to effectively train the pattern mining models. During the evaluation phase, we then derive features from the extracted patterns and apply them to a supervised student outcome prediction task as a measure of the quality of the patterns.

2.1 Unique Element Patterns Detector

We begin with the representation for each action and propose our pattern detection model for unique element sequence mining At the first stage, we use one-hot encoding to represent each action that was taken by students. After that, we use a one-dimensional CNN (i.e., convolving only over time), without bias weights, to extract patterns of action subsequences. We constrain the parameters of CNN filters to directly impose an interpretable, discrete structure on the weights. To predict future actions, we append a fully-connected layer and sigmoid function.

The crux of our approach to discovering interpretable patterns of specific actions is to force each row (corresponding to one step in a sequence) in the CNN filters to have only one parameter that is close to 1, while all others are close to 0. To achieve the desired weight structure, we applied regularization to CNN filter parameters as part of training.

In our method, the primary training objective is to minimize the binary cross-entropy loss for predicting future actions. To enforce discrete structure of the filters of CNN, we utilize regularization to force the sum of each row of the parameters of each CNN filter to 1, while most parameters are 0, thereby leaving only a single 1 corresponding to a single action. We split the approach into two steps. The first step is to ensure the sum of the parameters in each row is close to 1, by adding the loss:

$$L_{\text{r\_sum}} = \sum_{p=1}^{M} \sum_{n=1}^{k} \left(1 - \sum_{j=1}^{d} W_{pnj}^{3}\right)^2 \tag{1}$$

where W refers to weight of convolutional neural network, d is the number possible actions (i.e., the size of each one-hot encoded vector), k is the number of sequence steps in each CNN kernel (i.e., the length of pattern to learn), and M is the number of filters in the CNN (i.e., the number of patterns to learn).

The second step is to encourage parameters to go toward 0 via row-wise L1 loss, leaving only one parameter close to 1 to minimize the $L_{\text{r\_sum}}$ row sum loss.

$$L_{\text{r\_l1}} = \sum_{p=1}^{M} \sum_{n=1}^{k} \sum_{j=1}^{d} |W_{pnj}| \tag{2}$$

Finally, we optimize the following joint objective function during training:

$$L = L_{\text{prediction}} + \alpha L_{\text{r\_sum}} + \beta L_{\text{r\_l1}} \tag{3}$$

where L_r is the whole structured regularization loss, and α and β are coefficients for each regularization part, included to balance the contrasting minimization objectives of $L_{\text{r\_sum}}$ and $L_{\text{r\_l1}}$.

2.2 Multiple Concurrent Elements Pattern Detector

The limitation of the unique action detector is that it can only handle situations where each step in the sequence contains exactly one action. In some circumstances, each step contains many actions or events, such as when a student does several activities logged with the same timestamp. We extended our approach to handle this condition, following the model proposed in the previous section with different constraints. Specially, we force each CNN kernel weight to be either close to 0 or close to 1, ignoring the sum of all weights and thus allowing multiple actions per step. To achieve this goal, the regularization loss for each parameter is minimized when the parameter is either 0 or 1.

We operationalized this regularization goal via the following quadratic equation:

$$L_{\text{r\_m}} = \sum_{p=1}^{M} \sum_{n=1}^{k} \sum_{j=1}^{d} |W_{pnj}^{2} - W_{pnj}| \tag{4}$$

Overall, the objective function is:

$$L = L_{\text{prediction}} + \gamma L_{\text{r\_m}} \tag{5}$$

In $L_{\text{r\_m}}$, γ serves as a weight we tuned to ensure that the structured regularization loss $L_{\text{r\_m}}$ has the desired effect on the CNN weights without over-emphasizing regularization relative to the prediction loss.

The prediction loss for our multiple concurrent elements example is to minimize binary cross-entropy with logits loss, though other loss functions could be applied.

2.3 Warm-up Period

Table 1: Performance comparison (Pearson's r correlation coefficient between predicted and actual student grades) of our method versus CM-SPAM on two datasets. The EPM dataset has grades for five learning sessions (labeled 2–6), while OULAD has grades for seven courses (labeled A–G). Results without warm-up and structured regularization are provided as points of comparison, though the CNN filters without structured regularization are not interpretable.

	EPM					OULAD						
	Course 2	3	4	5	6	Course A	B	C	D	E	F	G
CM-SPAM [5]	-.050	.603	.139	.134	.333	.318	.341	.341	.440	.381	.381	.510
Without warm-up	**-.032**	.729	.425	.055	.430	.324	.414	.514	.433	.456	.394	.544
Our approach	-.092	**.792**	.432	**.227**	**.450**	.330	**.461**	**.532**	**.500**	.498	**.433**	**.563**
Traditional CNN	-.199	.672	**.518**	.209	.375	**.365**	.416	.547	.511	**.506**	.422	.550

Frequently, models learned a local optimum where regularization losses were immediately optimized. To avoid getting stuck at local optima of the objective, we introduced a "warm-up" period to stabilize training [17]. In our experiment, we trained the model without regularization loss for five epochs. Subsequently, we linearly increased the coefficient of the regularization loss over the course of ten epochs.

2.4 Features for Transfer Learning

After learning the set of predictive patterns, we evaluated the utility of learned patterns for a subsequent prediction task (i.e., transfer learning with the learned patterns). We froze the weights of the CNN, then applied the network to generate pattern features for each student's sequence of actions. Note that each sequence is typically much longer than the number of steps in each CNN kernel. Thus, we aggregated filter activations for each pattern by applying basic statistical calculations, including sum, standard deviation, max, min, skew, kurtosis, and different quantiles (10%, 30%, 50%, 70%, 90%).

We then concatenated all of these aggregated values of all extracted patterns to create feature vectors. As a means to judge the quality of the pattern features, we predicted students' learning outcomes with a random forest regression model [4].

3. EXPERIMENTS

In this section, we first introduce the details of two datasets and a baseline pattern discovery algorithm, against which we compare our proposed method (Table 1). We use visualization to examine the learned patterns, and compare transfer learning predictions of student outcomes via Pearson correlations. Finally, we discuss the convergence of our method.

3.1 Datasets

We work on two public datasets that contain learning behaviors of students represented by actions from different courses.

Educational Process Mining (EPM). The EPM dataset [19] contains sequential records of 100 students' activities during 6 laboratory sessions (5 of which have outcome labels) of the digital design course at the University of Genoa. Actions were logged in sequential order, such that each row represented a unique action taken by a student. We describe activities included in EPM dataset, including their frequency, in the Appendix.

Open University Learning Analytics Dataset (OULAD). The OULAD dataset [12] contains data about courses, students, actions of students, and their interactions with a virtual learning environment (VLE; specifically, Moodle) for seven courses, which started from either February or October. We merged multiple semesters of the same course because the patterns in the same courses should be relatively (if not exactly) consistent. The detail of interaction events included in the dataset shown in the Appendix (we merged some infrequent interactions into *other* category because the frequency of occurrence of these interactions was rare).

3.2 Baseline Comparison Method

Typical sequential pattern mining algorithms include those like CM-SPAM, GSP [18], PrefixSpan [8], and SPADE [25]. We use CM-SPAM as a baseline method here because it can easily find patterns of a specific length, which allows fair comparison to our proposed method.

CM-SPAM [5] is a sequential pattern mining algorithm based on Sequential PAttern Mining (SPAM; [3]). SPAM is a depth-first sequential frequent pattern search algorithm. CM-SPAM prunes the SPAM search space to improve computational complexity. We focused on mining patterns with the highest support and matched the length of patterns in our method, selecting 25 of the highest-support patterns to compare against the 25 patterns learned by our method.

3.3 Experimental Setup

We optimized CNN models with Adam [10] for 50 epochs. We tuned hyperparameters including learning rate, loss coefficients, and warm-up duration, based only on results in OULAD course A, to avoid over-fitting hyperparameters to the other six OULAD courses or the EPM dataset. Hyperparameters related to the structure of input and the model architecture we left fixed. Specifically, we convolved CNN filters of length 3 over subsequences of current events of length 5, with stride length 1, and predicted the next 1 event. Models had 25 convolution filters (patterns to learn). We concatenated convolution filter outputs and used a fully-connected layer with sigmoid activation for predicting the next action. We found that the model worked best with the learning rate set to .001, after testing .01, .0075, .005, .0025, and .001.

Components of the structured regularization loss have notably different magnitudes for the unique action case, since the L1 loss component ($L_{\text{r\_l1}}$) is several times larger than

the filter row sum component (L_{r_l1}). We thus applied a relatively large weight for L_{r_l1} and small weight for L_{r_l1} to balance regularization terms and achieve the desired weight structure. We tried different ratios of α/β, including 1, 10, 20, 30, 40, 50, 60, 70, 100, and 200. With $\alpha = 0.075$ and $\beta = 0.0075$, loss converged well. For the warm-up procedure, after testing 1 epochs, 5 epoch, and 10 epochs, we found the model converged best with $n_{total} = 5$ epochs.

For evaluating pattern utility via transfer learning with random forest regression, a higher correlation score (Pearson's r ranging from -1 to 1) represents patterns with higher utility for the downstream task.

3.4 Performance Analysis

Outcome prediction is a natural way to evaluate the utility of patterns [11]. We did so using the transfer learning approach described above, and split each course from each dataset into a train/test set at the student level with a ratio of 2:1 for evaluation.

3.4.1 Quantitative Results

The results of our method are shown in Table 1. Because students' learning actions contain meaningful sequential dependencies with each other, which patterns that happen frequently do not naturally capture. Consequently, CM-SPAM patterns were slightly less useful for inferring high level information (predicting student outcomes), as shown in Table 1. Additionally, instructors may be able to interpret the patterns extracted by our methods to intervene in future courses, given that the extracted patterns are few enough in number (25) to manually review and are related to outcomes. Generally, our approach outperformed traditional sequential pattern mining for almost all courses across the two datasets. Our approach improved the correlation of sequences with outcomes by an average of $r = .131$ on the EPM dataset, and was as good or better than CM-SPAM (r improved by .101) on the OULAD dataset. The result confirmed the usefulness of our predictive patterns derived via self-supervised learning.

3.4.2 Pattern Visualization and Analysis

We visualize the patterns extracted by our approach, demonstrate that they have the desired structure, and compare them with the traditional CNN filters.

To compare our method and typical CNN patterns, typical densely-distributed CNN weights lend little insight into the specific sequences of actions that activate filters (shown in Figure 1). These CNN patterns conflate selection of relevant input actions with weighting those patterns, which prevents their use as a sequence mining method. In addition, typical CNNs only extract patterns that correlate with student outcomes (in fully supervised applications). As a result, they do not necessarily learn dependencies among students actions; it remains to be seen whether our method applied in a fully-supervised model would produce notably different sequences of behaviors. Regardless, the patterns learned by our method are interpretable relative to typical CNN filter weights, which makes them straightforward to utilize for other downstream tasks even though they come from a self-supervised model. However, patterns with multiple concurrent actions are still less easily interpreted than patterns for unique actions given the possibility of many different concurrent actions. They are, however, still straightforward to transfer to downstream tasks.

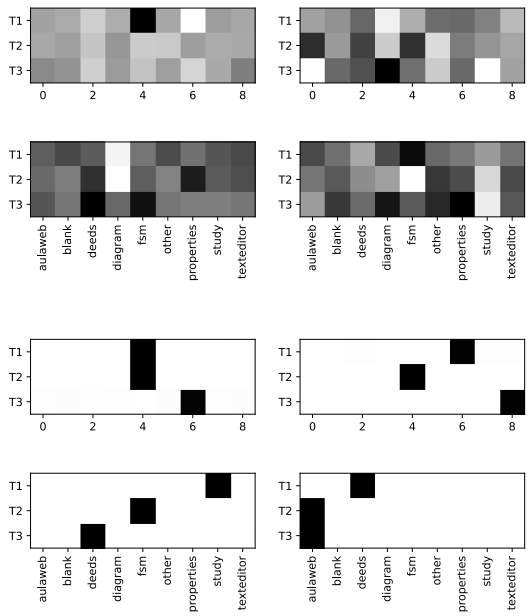


Figure 1: The bottom two rows are randomly-chosen example patterns extracted by our approach based on EPM course 2. The top two rows are traditional CNN filters.

4. CONCLUSION

In this paper, we presented a general self-supervised sequence mining algorithm that works for both sequences of individual actions and multiple concurrent actions. We mined sequential patterns by convolutional neural networks, and applied transfer learning to judge the quality of the extracted patterns for predicting student outcomes as an example downstream prediction task. Our results showed that the extracted patterns were indeed useful, as measured by the correlation between predictions and student outcomes.

We empirically demonstrated that the patterns extracted by our method have similar or higher utility for two prediction tasks than those extracted via a traditional frequent pattern mining algorithm, while the extracted patterns can still be easily interpreted. Furthermore, our approach deals with common pattern mining problems like pattern explosion by training a fixed number of convolutional filters, where filters are selected from the space of all possible filters via stochastic gradient descent.

In summary, our approach is a novel and interpretable way to extract predictive patterns of actions from sequential data.

5. REFERENCES

[1] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proceedings 20th International conference Very Large Data Bases,*

VLDB, volume 1215, pages 487–499, 1994.

[2] Ş. Aydoğdu. A new student modeling technique with convolutional neural networks: Learnerprints. *Journal of Educational Computing Research*, page 0735633120969216, 2020.

[3] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 429–435, 2002.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas. Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 40–52. Springer, 2014.

[6] M. N. Garofalakis, R. Rastogi, and K. Shim. Spirit: Sequential pattern mining with regular expression constraints. In *VLDB*, volume 99, pages 7–10, 1999.

[7] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 355–359, 2000.

[8] J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224. Citeseer, 2001.

[9] T. Hu, G. Sun, and Z. Xu. Assessing student contributions in wiki-based collaborative writing system. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, pages 615–619, 2020.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] S. Klingler, R. Wampfler, T. Käser, B. Solenthaler, and M. Gross. Efficient feature embeddings for student classification with variational auto-encoders. *International Educational Data Mining Society*, 2017.

[12] J. Kuzilek, M. Hlosta, and Z. Zdrahal. Open university learning analytics dataset. *Scientific Data*, 4:170171, 2017.

[13] M.-Y. Lin and S.-Y. Lee. Fast discovery of sequential patterns by memory indexing. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 150–160. Springer, 2002.

[14] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 81–88, 2001.

[15] S. Shen, Q. Liu, E. Chen, H. Wu, Z. Huang, W. Zhao, Y. Su, H. Ma, and S. Wang. Convolutional knowledge tracing: Modeling individualization in student learning process. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1857–1860, 2020.

[16] X. Shen, B. Yi, Z. Zhang, J. Shu, and H. Liu. Automatic recommendation technology for learning resources with convolutional neural network. In *2016 International Symposium on Educational Technology (ISET)*, pages 30–34. IEEE, 2016.

[17] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 3738–3746, 2016.

[18] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 1996.

[19] M. Vahdat, L. Oneto, D. Anguita, M. Funk, and M. Rauterberg. A learning analytics approach to correlate the academic achievements of students with interaction data from an educational simulator. In *Design for Teaching and Learning in a Networked World*, pages 352–366. Springer, 2015.

[20] H. Valdiviejas and N. Bosch. Using association rule mining to uncover rarely occurring relationships in two university online stem courses: A comparative analysis. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, pages 686–690, 2020.

[21] M. Van Leeuwen. Interactive data exploration using pattern mining. In *Interactive knowledge discovery and data mining in biomedical informatics*, pages 169–182. Springer, 2014.

[22] R. Wampfler, A. Emch, B. Solenthaler, and M. Gross. Image reconstruction of tablet front camera recordings in educational settings. *International Educational Data Mining Society*, 2020.

[23] J. Wang and J. Han. Bide: Efficient mining of frequent closed sequences. In *Proceedings 20th International Conference on Data Engineering*, pages 79–90. IEEE, 2004.

[24] Y. Xiao, G. Zingle, Q. Jia, H. R. Shah, Y. Zhang, T. Li, M. Karovaliya, W. Zhao, Y. Song, J. Ji, et al. Detecting problem statements in peer assessments. *arXiv preprint arXiv:2006.04532*, 2020.

[25] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1-2):31–60, 2001.

6. APPENDIX: DATASET DETAILS

Table 2: Description of actions in the OULAD dataset. Infrequent actions were grouped together into an *other* category, with the exception of *transfer* given that it is one of the most semantically important, along with *register* and *unregister*.

Action	Description	Frequency
homepage	Visit the main course page	1,735,226
gap	One or more consecutive days with no action	860,356
oucontent	View course content page	829,476
forumng	Discussion forum usage	822,895
subpage	Manage/view course activities on a page other than the homepage	804,577
resource	Download a document from the course	399,961
url	Click a link to an external site	314,240
quiz	Take a quiz	211,497
exam	Take an assessment	160,498
ouwiki	Access the course wiki	89,406
register	Register for the course	32,548
unregister	Drop the course	10,072
transfer	Transfer grade from previous session (semester)	526
	Infrequent activities grouped together as "other"	
page	Non-interactive information page	47,549
oucollaborate	Audio/video conferencing	47,334
externalquiz	Externally-hosted quiz	41,642
glossary	View course glossary	17,258
questionnaire	Access survey form	15,109
ouelluminate	Audio-only conferencing	11,384
dualpane	Side-by-side view of instructions and related content	9,256
dataplus	Interact with a toy SQLite database	6,818
htmlactivity	Interactive HTML page	6,016
folder	View folder containing related activities	4,678
sharedsubpage	View page shared from another course	148
repeatactivity	Activities repeated from earlier in the course	3

Table 3: Description of activities in the EPM dataset. Internal activity names from the EPM dataset are provided to enable unambiguous matching to the original data.

Action	Description	Frequency
texteditor	Use a text editor	42,431
deeds	Other DEEDS (Digital Electronics Education and Design Suite) activities	38,372
other	Not viewing any pages described above (mostly off-task activities)	33,602
blank	Title of visited page is not recorded	24,303
study	View exercises or materials related to courses	22,261
diagram	Use a "simulation timing diagram" to test a solution	20,815
fsm	Use a finite state machine (FSM) simulator	20,596
properties	Set parameters of a simulation or design	19,677
aulaweb	Visit learning management system	8,261

Restructuring Curricular Patterns Using Bayesian Networks

Ahmad Slim
Lebanese American University
1102, Beirut, Lebanon
ahmad.slim@lau.edu.lb

Gregory L. Heileman
The University of Arizona
Tucson, AZ 85721, U.S.
heileman@arizona.edu

Chaouki T. Abdallah
Georgia Tech
Atlanta, GA 30332, U.S.
ctabdallah@gatech.edu

Ameer Slim
The University of New Mexico
Albuquerque, NM 87131, U.S.
ahs1993@unm.edu

Najem Sirhan
The University of New Mexico
Albuquerque, NM 87131, U.S.
najem83@unm.edu

ABSTRACT

Recent studies proved the existence of a relationship between the complexity of university curricula and graduation rates. As a result, extensive efforts have been done in an attempt to restructure curricula in order to improve graduation rates. In this paper, we propose a new model for evaluating and quantifying the impact of restructuring curricula on graduation rates using a Bayesian network framework. We validate our model by analyzing a common curricular pattern found in most of the engineering programs. We demonstrate its usefulness using actual data for students at the University of New Mexico. We also extend this model to include a helpful tool that can be used to predict student performance. The advantage of our work is characterized by its data-driven nature which makes it more reliable than other proposed models.

Keywords

Curricular analytics, Bayesian networks, education, curriculum complexity, student success, graduation rate

1. INTRODUCTION

Recently a significant amount of work has been done on curricular analytics to show its impact on student success [16, 15, 12, 13, 2, 14]. The work done mainly spots the light on the importance of curricula structure on student performance characterized primarily by graduation and retention rates. These studies argue that the complexity of prerequisite dependencies between the requirements of a curriculum can increase the risk of students stopping out and eventually dropping the school. A lot of work has been done recently identifying factors that help students retain their school and hence graduate at faster rates [6]. This includes new learning pedagogy styles, dorms, flipped classrooms, learning centers, etc. However, such factors solely might fail to significantly contribute to student success if other institutional factors are overlooked; essentially curricula structure [11]. As mentioned earlier, it is already proved that the structure of a curriculum has a direct impact in student success [2, 3]. In this regard, Klingbeil and Bourne considered a case study and analyzed the structure of a common curricular pattern found in most of the engineering programs (Figure 1) [5]. They noticed that, in the sophomore year, most of the engineering programs require Differential Equations as a prerequisite to a domain specific course (in electrical engineering programs, Circuits I is domain specific; in mechanical engineering, Mechanics (statics and dynamics) is domain specific; etc.). They also noticed that all the learning outcomes in Differential Equations, except for solving linear differential equations, are not necessarily required to pass the domain specific course. Thus they suggested to create a new course in the freshman year to teach how to solve linear differential equations along with the Precalculus materials. They called this new course Engineering 101. As a result of this observation, they pointed out that Differential Equations is not required anymore as a prerequisite for the domain central course; only Engineering 101 does. This resulted in a revised curricular pattern shown in Figure 2. Klingbeil and Bourne claim that this new curricular pattern will help students graduate in a timely fashion. This is driven by the fact that the students are not required anymore to follow the long chain of prerequisites before they are allowed to take a domain specific course-as it is the case in the original curricular pattern. This new pattern is now pursued by a number of universities [5]. And to validate the legitimacy of such changes to curricular patterns, a number of researchers came up with different mathematical models that prove the significance of these changes on student success outcomes. Slim et al. came up with a metric that quantifies the complexity of any curricular pattern [16]. Their metric showed that the revised engineering pattern shown in Figure 2 is less complex than that of the original one shown in Figure 1[2]. Thus, according to their metric, students are expected to finish their degrees at a faster pace. Furthermore, Hickeman showed that the revised engineering pattern can significantly improve graduation rates [3]. He proved that by implementing a Monte Carlo simulation through which virtual students are allowed to flow through a curricular pattern. For more details on different models see [2]. Although these models put a mathematical foundation to prove the advantage of restructuring curricula, they still have a major limitation. None of these models include actual student data in their implementations. In other words, these models only show the advantage of restructuring curricular patterns in its abstract state without using any data-driven approaches. This makes the models less reliable in proving the need to revise any curricular pattern. In this paper we propose a data-driven model that uses actual student data to achieve this. Particularly we use a Bayesian Network (BN) model to statistically prove the validity of any effort to restructure

Ahmad Slim, Gregory Heileman, Chaouki Abdallah, Ameer Slim and Najem Sirhan "Restructuring Curricular Patterns Using Bayesian Networks". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 767-770. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

a curricular pattern. This is mainly characterized by adding and removing courses/prerequisites within a curricular pattern which can be neatly captured using a BN. The main motivation behind our proposed model is the ability to use the notion of hidden/latent variables in building a BN. These hidden nodes represent new added courses in the revised curricular pattern. Thus they can be used to check the validity of the changes made to the original curricular pattern and accordingly decide whether to apply these changes or not. It is important to note that our model can be generalized to find the optimal structure of a revised curricular pattern using different methods of structural learning for a BN [1]. However we will not cover this part in our paper. We will leave it for a future work. The remainder of this paper is structured as follows. In Section II we present the details of our proposed framework and provide a case study. In Section III we present a number of applications for our proposed model and show some simulation results. Finally, Section IV presents some concluding remarks.

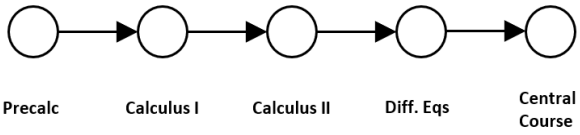


Figure 1: The original curricular pattern found in most engineering programs. The nodes represent the courses and the directed edges represent the prerequisite dependencies.

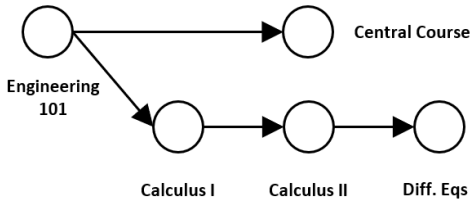


Figure 2: The revised curricular pattern.

2. BAYESIAN NETWORKS AND HIDDEN VARIABLES

A BN is a directed acyclic graph (DAG) representing correlations between a number of random variables [4]. The graph-like structure reflects a confined representation of the joint probability distribution underlying these variables. The presence of an edge between two nodes indicates the existence of a relationship between two variables and the direction of the edge indicates the direction of the causal relationship. That is a directed edge from node A to node B indicates that A causes B. In BNs, these types of relationships are quantified by conditional probability tables (CPTs). The main feature of a BN is its compact representation of the conditional dependencies of the random variables. However, in some applications the BN gets complicated and thus in this case adding hidden nodes would be essential for two main reasons [8]:

1. **Knowledge discovery**: reveals interesting relationships among the variables of the data
2. **Lower complexity**: attains lower structure complexity of the network

For example, consider the case where we observe a bunch of variables representing different patient's symptoms. The joint probability distribution for these symptoms might be highly connected. In this case, the BN representation of these symptoms would be highly

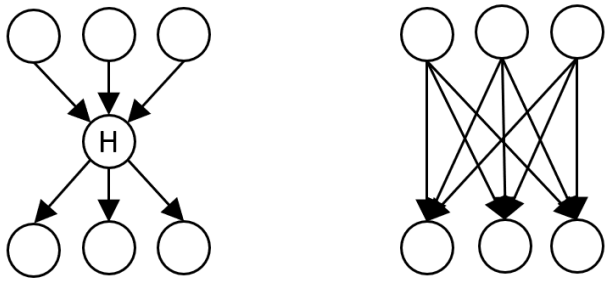


Figure 3: Two DAGs (one with a hidden node and the other without) representing the relationship between symptoms, causes and mediating factors. Symptoms, such as chest pain, are represented by the leaves. Causes, such as smoking and diet, are represented by the roots. Mediating factors, such as heart disease, are represented by the hidden nodes. This figure shows how hidden nodes can reveal a better understanding of the relationship between variables by attaining a lower structure complexity of the network.

complicated. However if we introduce a "cause" node representing the underlying disease for these symptoms then we can get a noticeably simpler network. We call this cause node a hidden node. Figure 3 (inspired from [7]) illustrates this example in more details. In a similar scenario in an educational context, a curricular pattern can be modeled as a BN. A node maybe a course, and the states of the node would be the possible letter grades (i.e., $A, B+, B, C+$, etc.). A directed edge from course A to course B indicates that the performance in A influences that in B. In this context adding a hidden node to the BN is equivalent to adding a new course to a curricular pattern. This hidden node might represent the underlying prerequisite course that needs to be taken prior taking other courses. This notion constitutes the bulk of our proposed work and the subsequent sections elaborate more about this idea. Following this notion, restructuring the BN of any curricular pattern would include these steps:

- Removing an existing course(s) (i.e., removing a node)
- Adding a new course(s) (i.e., adding a hidden node)
- Removing an existing prerequisite(s) (i.e., removing a directed edge)
- Adding a new prerequisite(s) (i.e., adding a directed edge)

We denote the restructured BN by R, characterizing the revised curricular pattern. Once R is constructed, we fit the CPTs using actual student data, denoted by D, and then compute the likelihood of R, $p(D/R)$. Similarly, we denote by O the BN of the original curricular pattern. Once it is constructed, we compute its likelihood, $p(D/O)$. If $p(D/R)$ is greater than $p(D/O)$, then the revised version of the curricular pattern fits the student data better than that of the original one. In this case, the proposed revised curricular pattern can be a good candidate to replace the original one. To elaborate more about this, we present a case study in the following section.

2.1 Engineering Curricular Pattern: A Case Study

In this section, we consider, as a case study, the engineering curricular patterns shown in Figure 1 and Figure 2. Recall that the graph in Figure 1 represents the original pattern whereas that in Figure 2 represents the revised one. For these two graphs, we construct two BNs denoted by O and R respectively. These two BNs are shown in

Course Number	Course Name
MATH 150	Precalculus
MATH 162	Calculus I
MATH 163	Calculus II
MATH 264	Calculus III
MATH 316	Differential Equations
PHYC 160	General Physics I
PHYC 161	General Physics II
ECE 203	Circuits I
ENG 101	New Course Proposed

Table 1: Engineering courses taught at freshman and sophomore level at UNM.

Figure 4 and Figure 5. The variables used to construct O and R are shown in Table 1. These variables represent actual courses taught at the University of New Mexico (UNM). The states of each of these variables are the letter grades: A, B, C and D/F where D and F are assumed to represent one state. The CPTs for both O and R are computed using a dataset, denoted by D, for $1,000$ UNM student. Each row in this dataset contains the letter grades achieved in the courses shown in Table 1. Some grades for some students were not available. Thus, the dataset included missing values. In addition, ENG 101 in Figure 5 is considered a hidden variable because it is supposed to represent the new proposed course and thus we do not have the letter grade values. Therefore, to compensate for hidden and missing values in our dataset, we used the expectation-maximization (EM) method to compute the CPTs for O and R [9]. Using EM, we computed the ratio of the likelihood, $\frac{p(D/R)}{p(D/O)}$, to be 2.89. This means that R fits the student data better than O. This result suggests that the proposed revised curricular pattern is a good candidate to replace the original one. Not only does it have a less complex structure but also the revised curricular pattern has the potential to improve student performance when compared to the original pattern. We concluded this using actual student data which is an advantage over other proposed models in literature [2]. In the following sections we present more applications of BNs in the context of course network.

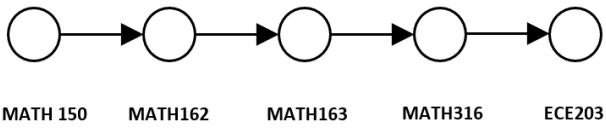


Figure 4: The original curricular pattern found in the electrical engineering curriculum.

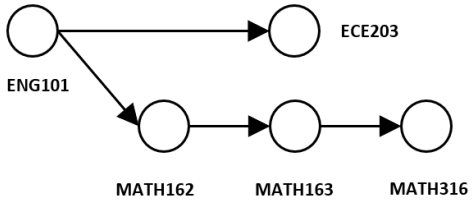


Figure 5: The revised version of the curricular pattern in the electrical engineering curriculum.

3. INFLUENCE OF STUDENT CHARACTERISTICS ON ACADEMIC PERFORMANCE

As mentioned earlier, many institutions are dedicating lot of efforts on student success. Colleges and universities are applying ever more sophisticated analytical tools to track their student progress in an attempt to improve their performance characterized mainly by graduation and retention rates. Intuitively, early indicators of student performance in this context is crucial to provide suitable interventions when needed. Thus predicting the performance of students in future courses is essential to achieve it. In this regard, historical information about previous academic achievement of a student could be used to project future performance. For example, a student who receives a 'B' in Calculus I is expected to receive a better grade in Calculus II compared to those who receive a 'D'. A BN in the context of course network can capture the correlation in performance between such courses. Further, it can be used to predict the letter grades of a student in subsequent classes based on the grades of previous classes. The accuracy of prediction can further be improved by adding other factors related to student characteristics. Factors such as age, gender, high school GPA, socioeconomic status, etc. proved to influence student performance [11]. For this reason, it would be rational to add such factors as additional variables to the BN of a course network. The advantage of using a BN model to capture all these variables together is two-folded: it can be used as a knowledge discovery model that can neatly display the correlation between the variables and also it can be used as an inference tool to predict student performance. In this section, we construct a BN for eight engineering courses along with five other variables representing student characteristics. The eight courses are: MATH 150, MATH 162, MATH 163, MATH 264, MATH 316, PHYC 160, PHYC 161 and ECE 203 (Table 1). These courses are considered to be the most crucial classes at the freshman level. As for student characteristics, we considered Gender, ACT score, and high school GPA. As mentioned earlier the student characteristics are proved to influence student performance. Thus it would be interesting to discover and visualize how all these variables are related to each other. In the following section we show the constructed BN along with some applications. However, it is important to mention here that a similar work has been done in [10]. The authors of this work used a domain expert to construct the BN for these variables. This means that the process of constructing the BN is not automated and doesn't guarantee a good fit to the student data. In this paper, however, we automated the process of learning the structure of the BN using a score-based learning algorithm [1]. Particularly, we used the hill climbing (hc) greedy search that explores the space of the DAGs by single-arc addition, removal and reversals.

3.1 A BN for Engineering Courses

To construct and validate our framework, we collected a dataset of 3,000 undergraduate student in the college of engineering at UNM. The dataset represented different demographical and academical variables of the students. The states for each of these variables are show in Table 2. It is important to note that MATH 162 is the prerequisite for MATH 163, MATH 163 is the prerequisite for MATH 316 and MATH 264, and MATH 316 is the prerequisite for ECE 203. The constructed BN is presented in the graph shown in Figure 6. It is tempting to interpret this graph in terms of causality. In particular, it seems that ACT score, high school GPA and gender, in contrast to ethnicity, causally influence the performance of students in these engineering courses. Also, this graph shows that the performance in PHYC 160 influences that in MATH 264 and ECE 203. This is an interesting observation because, according to the department policy, PHYC 160 is not a required prerequisite for neither of these courses. Though it has an impact on both these courses. Another interesting observation is the absence of any correlation in performance between MATH 316 and ECE 203 even though MATH 316 is a prerequisite to ECE 203. This observation confirms the fact that

Variable	States
Gender	Female, Male
Ethnicity	7 different ethnicities
ACT score	Integer between 10 and 36
High school GPA	Real value between 1 and 4
MATH 150,162,163,264,316	A,B,C, D/F
PHYC 160,161	A,B,C, D/F
ECE 203	A,B,C, D/F

Table 2: The courses and the student characteristics used to build the BN.

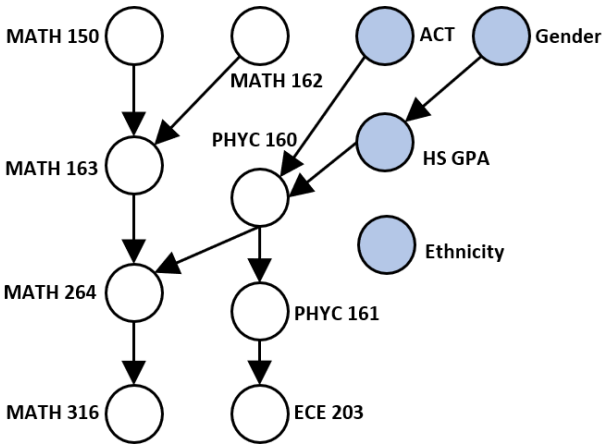


Figure 6: The constructed BN using UNM student data.

only a small portion of the learning outcomes in MATH 316, namely the ability to solve linear differential equations, are actually used in ECE 203 and the absence of a link between these two courses proves it. This is another evidence that supports the claim that it is needed to restructure the original curricular pattern shown in Figure 1.

4. CONCLUSION

In this paper we presented a framework that models a curriculum as a Bayesian Network (BN). We showed that this model, in the context of a course network, can be used to quantify any effort to restructure curricular patterns. In particular, we used the notion of hidden variables to achieve this. We validated our proposed model using a common curricular pattern found in most of the engineering programs. For that, we used actual data for students at the University of New Mexico. The results showed that the likelihood of the revised version of the engineering curricular pattern is higher than that of the original one. This suggests that the revised version can help students perform better in their courses as well as graduate at a faster pace. The advantage of our model over other proposed models in literature is its data-driven nature which makes it more reliable. Furthermore, we extended our model to use it as a knowledge discovery and inference tool. Particularly we added variables related to student characteristics (e.g. gender, ACT score, high school GPA, etc.) and showed how they can influence student performance. We also showed how to exploit the constructed BN to predict the grades of a student in following semesters based on grades of previous semesters.

5. REFERENCES

[1] D. Heckerman. Learning in graphical models. chapter A Tutorial on Learning with Bayesian Networks, pages 301–354. MIT Press, Cambridge, MA, USA, 1999.

[2] G. L. Heileman, C. T. Abdallah, A. Slim, and M. Hickman. Curricular analytics: A framework for quantifying the impact of curricular reforms and pedagogical innovations. *CoRR*, abs/1811.09676, 2018.

[3] M. Hickman. Development of a Curriculum Analysis and Simulation Library with Applications in Curricular Analytics. Master's thesis, The University of New Mexico, 2017.

[4] M. Horny. Bayesian networks. Technical report, Boston University School of Public Health, 2014.

[5] N. W. Klingbeil and A. Bourne. The wright state model for engineering mathematics education: Longitudinal impact on initially underprepared students. In *2015 ASEE Annual Conference & Exposition*, Seattle, Washington, June 2015. ASEE Conferences.

[6] L. K. Lau. Institutional factors affecting student retention. 2003.

[7] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[8] T. L. Perez and L. Kaelbling. Techniques in artificial intelligence (sma 5504). Massachusetts Institute of Technology, MIT OpenCourseWare, Fall 2002.

[9] D. Prescher. A tutorial on the expectation maximization algorithm including maximum likelihood estimation and em training of probabilistic context free grammars. *ArXiv*, 2004.

[10] A. Sharabiani, F. Karim, A. Sharabiani, M. Atanasov, and H. Darabi. An enhanced bayesian network model for prediction of students' academic performance in engineering programs. *2014 IEEE Global Engineering Education Conference (EDUCON)*, pages 832–837, 2014.

[11] A. Slim. *Curricular Analytics in Higher Education*. PhD thesis, The University of New Mexico, 2016.

[12] A. Slim, G. L. Heileman, W. Al-Doroubi, and C. T. Abdallah. The impact of course enrollment sequences on student success. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 59–65, March 2016.

[13] A. Slim, G. L. Heileman, M. Hickman, and C. T. Abdallah. A geometric distributed probabilistic model to predict graduation rates. In *2017 IEEE Cloud Big Data Computing (CBDCom)*, pages 1–8, Aug 2017.

[14] A. Slim, D. Hush, T. Ojha, , C. T. Abdallah, G. L. Heileman, and G. El-Howayek. An automated framework to recommend a suitable academic program, course and instructor. In *Proceedings of the Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, San Francisco, CA, USA, 2019. IEEE.

[15] A. Slim, J. Kozlick, G. L. Heileman, and C. T. Abdallah. The complexity of university curricula according to course cruciality. In *Proceedings of the 8th International Conference on Complex, Intelligent, and Software Intensive Systems*, Birmingham City University, Birmingham, UK, 2014. IEEE.

[16] A. Slim, J. Kozlick, G. L. Heileman, J. Wigdahl, and C. T. Abdallah. Network analysis of university courses. In *Proceedings of the 6th Annual Workshop on Simplifying Complex Networks for Practitioners*, Seoul, Korea, 2014. ACM.

Towards automated content analysis of feedback: A multi-language study

Ikenna Osakwe[1], Guanliang Chen[1], Alex Whitelock-Wainwright[1], Dragan Gašević[1], Anderson Pinheiro Cavalcanti[2], and Rafael Ferreira Mello[2]

{richard.osakwe, guanliang.chen, alex.wainwright, dragan.gasevic}@monash.edu

apc@cin.ufpe.br, rafael.mello@ufrpe.br

[1]Monash University, [2]Universidade Federal Rural de Pernambuco

ABSTRACT

Feedback is a crucial element of a student's learning process. It enables students to identify weaknesses and improve self-regulation. However, studies show this to be an area of great dissatisfaction in higher education. With ever-growing course participation numbers, delivering effective feedback is becoming an increasingly challenging task. Hence, this paper explores the use of automated content analysis to examine feedback provided by instructors for good feedback practices measured on *self*, *task*, *process*, and *self-regulation* levels. For this purpose, four binary XGBoost classifiers were trained and evaluated, one for each level of feedback. The results indicate effective classification performance on self, task, and process levels with accuracy values of 0.87, 0.82, and 0.69, respectively. Additionally, inter-language transferability of feedback features is measured using cross-language classification performance and feature importance analysis. Findings indicate a low generalizability of features between English and Portuguese feedback spaces.

1. INTRODUCTION

Despite widespread recognition of feedback's importance to learning [23, 29, 10], much of the current literature indicates a pervasiveness of low quality feedback in higher education [13]. Feedback quality is consistently rated one of the greatest causes of dissatisfaction for higher education students [9]. LA researchers are actively exploring automated feedback solutions that can enable instructors to efficiently identify and employ good feedback practices, and improve the speed of feedback delivery to students [15]. In that vein, several studies [17, 19, 28, 30] have examined the use of data mining methods to generate automated textual feedback. These analyses are often limited to domain specific areas such as computer programming or writing, or lack of grounding in educational theory. Much less work has gone into the exploration of automated domain-agnostic analysis to identify good feedback practices [4, 24]. Progress in such areas can enhance the instructor's ability to provide effective feedback comments and analyze features associated with good feedback practices for generalizable feedback generators. Therefore, this study aims to answer the following Research Questions (RQs):

1. To what extent can the automated analysis of feedback messages be used to identify good feedback practices?
   (a) How accurate are the predictions that are made about these feedback practices?
   (b) What are specific features of text that can be used to predict the use of good feedback practices?
2. How transferable are the identified feedback features to text written in different languages?

2. METHOD

2.1 Data

The dataset used in the current study consisted of feedback comments provided by instructors in Learning Analytics, Software Engineering, and Environmental Studies courses. A total of 2,092 observations were taken; 1,000 Portuguese records and 1,092 English records.

2.2 Coding Scheme

This study utilized Hattie and Timperley's [14] four levels of feedback due to its suitability for textual analysis due to its focus learning tasks, learning process, and self-regulation Cavalcanti et al. [4]. Hence, feedback examples were coded using Hattie and Timperley's [14] proposed four levels of feedback (see Table 1).

Feedback examples were coded by experts using instructions of Hattie and Timperley's [14] study. Each feedback record was examined by two expert coders separately. After this step, the differences between each pair of experts were compared. For the Portuguese feedback examples, the inter-rater agreement reached 72.2% with a Cohen's kappa (inter-rater ability considering chance [7]) of 0.44. The English feedback comments had inter-rater agreement of 63.8% and Cohen's kappa of 0.38. These measures met expectations for content analysis experimentation [20].

Ikenna Osakwe, Alexander Whitelock-Wainwright, Guanliang Chen, Rafael Ferreira Mello, Anderson Pinheiro Cavalcanti and Dragan Gašević "Towards automated content analysis of feedback: A multi-language study". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 771-776. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Table 1: Four levels of feedback identified by Hattie and Timperley [14]. Each level specifies different elements that the feedback is targeting and can be regarded as hierarchical, ranging from general comments made about the student themselves up to directives on how to improve self-regulation.

Level	Description	Example
Feedback about the self (FS)	Personal evaluations about the learner	"You are a bright student"
Feedback about the task (FT)	How well tasks are understood or performed	"You need to include more about the Treaty of Versailles."
Feedback about the process (FP)	Processes needed to understand or perform tasks	"You need to edit this piece of writing by attending to the descriptors you have used so the reader is able to understand the nuances of your meaning."
Feedback about self-regulation (FR)	How to improve self-regulation	"You already know the key features of the opening of an argument. Check to see whether you have incorporated them in your first paragraph."

Table 2: Number of instances for each class in the training and test datasets for each level of feedback.

		Class 0	Class 1	Total
FS	**Train**	1149 (82.19%)	249 (17.81%)	1398 (70%)
	Test	567 (82.17%)	123 (17.83%)	690 (30%)
FT	**Train**	602 (43.06%)	796 (56.94%)	1398 (70%)
	Test	297 (43.04%)	393 (56.96%)	690 (30%)
FR	**Train**	1290 (92.27%)	108 (7.73%)	1398 (70%)
	Test	637 (92.32%)	53 (7.68%)	690 (30%)
FP	**Train**	808 (57.80%)	590 (42.20%)	1398 (70%)
	Test	399 (57.83%)	291 (42.17%)	690 (30%)

The annotation process led to a dataset with four sets of binary classes: class 0 if a feedback message did not belong to a particular level; class 1 if the feedback message belonged to the feedback level.

2.3 Feature Engineering

Feature extraction was informed by relevant studies [4, 24, 16]. The studies promote the use of linguistic features such as those developed in LIWC (Linguistic Inquiry and Word Count) [27] and Coh-Metrix [11] over traditional textual features such as lexical N-grams or Part-Of-Speech. According to Kovanović et al. [16], these features encourage overfitting by inflating the feature space. Additionally, these traditional features are data dependent and thus make it difficult to define the feature space beforehand [16]. Hence, we used feature sets that incorporated 86 LIWC [27] features, 78 Coh-Metrix [11] features, and two additional features, which are relevant to this content area — number of named entities and language of delivered feedback.

2.4 Analysis

2.4.1 Data Analysis and Pre-processing

For the general classifier, feedback examples from both the English and Portuguese datasets were combined and split into 70% training and 30% test sets (Table 2). The training data suffered from class imbalances; particularly at the FS and FR levels.

2.4.2 Handling Class Imbalance

Studies have shown class imbalances can have a negative impact on model prediction performance [26]. To alleviate the class imbalance problem, sampling algorithms are often employed to adjust the ratio of represented classes. SMOTE is a popular oversampling method that analyzes the data records in a two-dimensional vector space of given classes and generates data points as a linear combination of existing data points [5].

2.5 Model Selection and Evaluation — RQ1a

Decision tree ensembles are widely regarded classification algorithms that are well suited to feedback analysis [4, 24]. This is due to their white-box properties, easy interpretability, high accuracy and ability to identify important features in a dataset [4, 24, 6, 8].

This study employed a decision tree implementation called XGBoost [6]. XGBoost has been shown to outperform Random Forest on numerous classification tasks [22, 31]. The algorithm utilizes gradient boosting, which involves sequentially combining models (in this case, decision trees) that predict the residuals or errors of previous models at each iteration to improve overall accuracy [6]. XGBoost is ideal due to their superior accuracy and their implicit analysis of feature importance [6]. Four binary XGBoost classifiers were trained; one for each level of feedback.

2.5.1 Feature Analysis —- RQ1b

The outputs of decision tree models can be analyzed with tools such as SHAP (SHapley Additive exPlanations) [18].Given an input of a machine learning model and data records, SHAP leverages the concept of Shapley values by measuring the average marginal contribution of a feature over all possible permutations. SHAP can diagnose the most impactful features using their SHAP value, which is the mean absolute contribution of each feature [18]. A higher SHAP value for a feature implies a greater importance compared to another feature.

2.5.2 Feature Transferability — RQ2

To measure the transferability of features across languages, the dataset was split by language, creating Portuguese and English feedback datasets. Each of these datasets was split into training and test splits (70% training and 30% test set), and binary classifiers were trained and tuned, resulting in English feedback trained classifiers, and Portuguese feedback trained classifiers for each level of feedback, with the exception of the FR level. For the Portuguese feedback examples, the FR level had just eight positive instances out

Table 3: Performance of the classifiers trained to address research question RQ1 on the combined dataset involving both the English and Portuguese datasets. Legend: ACC – Accuracy; K – Cohen's kappa; F1 - F1 Score.

	FS			FT			FR			FP		
Class Balancing	ACC	K	F1	ACC	K	F1	ACC	K	F1	ACC	K	F1
None	**0.88**	**0.52**	**0.58**	**0.82**	0.64	**0.83**	**0.92**	0.00	0.00	0.68	0.33	0.57
SMOTE	0.87	0.51	**0.58**	**0.82**	**0.65**	**0.83**	0.91	0.04	0.07	**0.69**	**0.35**	**0.59**

of 1,000 records, which was not enough to train a machine learning algorithm [12]; hence, this level of feedback was excluded from all transferability analysis.

Once the English and Portuguese trained classifiers were developed, feature transferability was measured by i) *the inter-language prediction performance*: the prediction performance (measured by accuracy, F1 score and Cohen's κ) of the English trained classifier on the English test set was compared to the predictor performance on the Portuguese test set for the FS, FT, and FP levels of feedback. The same process was repeated for the Portuguese trained classifier; ii) *A comparison of significant features*: The most important features for the English and Portuguese trained classifiers are compared at the FS, FT and FP levels of feedback.

3. RESULTS AND DISCUSSION

The goal of this study was to examine how accurately one can model the feature space of good feedback practices, and how this feature space varies across languages. In that vein, four research questions were answered using novel statistical learning methodologies, with a view of promoting good feedback practices at scale.

3.1 Model Performance —- RQ1a

Research question RQ1 focused on investigating the extent to which the automated analysis of feedback messages can be used to identify good feedback practices. Four binary classifiers were developed using a variety of features (see 2.3) The best performing models were effective in identifying FS, FT and FP. While not a direct comparison due to the addition of the English feedback examples, the models achieved better results over those reported by Cavalcanti et al. [4]. The classifiers were able to improve accuracy by 0.07 and 0.05 for FT and FP, respectively and increase kappa values by 0.11, 0.35, and 0.06 for FS, FT, and, FP, respectively.

Similar to previous works [4], the FR classifier was not as effective in identifying instances. The model obtained a poor kappa of 0.06, which was likely caused by the model's poor ability of detecting positive cases of FR. Poor performance on this level was due to the significantly lower cases of positive instances as compared to the other levels of feedback.

3.2 Feature Analysis —- RQ1b

The focus of research question RQ1b was analyzing the most important textual features associated with the four levels of feedback. Hattie and Timperley [14] state that FS involves evaluations of the person, which are often a form of praise. The current findings add weight to this claim, as those features found to be most important in predicting the FS level were affective processes (particularly, positive emotions) and social processes, which align with the concept of praise. FS is often thought to be the least effective level of feedback [3, 14] and relatedly, the FS classifier had a negative association with discrepancy words; this might indicate FS comments have little actionable information or insight.

FT is sometimes referred to as corrective feedback and provides information on details related to task accomplishment such as correctness or behavior. Accordingly, this study found the predictors most associated with FT were those that related to the amount of information provided. Specifically, higher values of word counts, frequency of content words and minimum frequency of content —- all of which can be linked to greater information —- were positively correlated with observance of FT. Hattie and Timperley [14] suggest instructors not to rely solely on FT, but rather to view it as a process that moves the student to FP and FR. This theory is backed by the finding of strong negative association of causation words and FT; hence, FT comments were less likely to illustrate the causes of the student's failings, which is essential for the learner's self-regulation [14, 3, 21].

Compared to FT, FP is believed to promote a deeper understanding of learning as it enables the identification of relationships between resources and output, and the development of stronger cognitive processes. To achieve this, Balzer et al. [1] state FP should concern information about actual relations in the learning environment, relations which have been recognized by the learner, and relations between the learning environment and the learner's perceptions. Therefore, the value of FP comes from providing useful information on relationships. The findings of this study corroborate the theoretical views of FP. Amongst the most important features for FP were frequency of content words, adverbs, negative connectives and discrepancy words. These imply that FP comments were tied to providing new and corrective information. Other significant features can be tied back to relationships; including frequency of semicolons (semicolons are often used to link together ideas) and features associated with space and relativity.

According to Butler [3], one of the goals of FR should be to improve the student's ability to monitor current progress and use that information to form effective learning strategies. Accordingly, some of the most important predictors of FR were greater present and future focused processes.

3.3 Feature Transferability -— RQ2

To address research question RQ2, we studied inter-language classifier performance, and compared the most significant features for classifiers trained on different language feedback. Barbosa et al. [2] used similar linguistic features to those used in this project, such as LIWC and Coh-Metrix, to study cross-language classification of cognitive presence in online discussions, and found features to be independent of language; hence, we expected to find a moderate level of generalizability of feedback features across languages. However, our findings indicate a low transferability of feedback features. As seen on Table 5, the average accuracy differential on inter-language performance amounted to -0.06, -0.59, and -0.26; while the average kappa differential was approximately -0.50, -0.27, and -0.33 for FS, FT, and FP, respectively. Likewise, the Portuguese and English trained classifiers showed minimal overlap in their most important

Table 4: Top 10 important features are measured using SHAP and displayed from most to least important for FS, FT, FR and FP classifiers.

FS			FT		
Variable	**Description**	**SHAP**	**Variable**	**Description**	**SHAP**
liwc.Exclam	Freq. of exclamation marks	1.02	cm.WRDFRQa	Freq. of all words	0.46
liwc.posemo	Freq. of words with positive emotion	0.73	cm.WRDFRQc	Freq. of content words	0.39
liwc.you	Freq. of the word "you"	0.24	cm.WRDFRQmc	Minimum freq. of content words	0.34
liwc.affect	Freq. of affective words	0.20	cm.DRNP	Noun phrase density	0.10
cm.SYNMEDlem	Minimal edit distance of lemmas	0.20	cm.DRAP	Adverbial phrase density	0.10
cm.WRDFRQc	Freq. of content words	0.15	liwc.SemiC	Freq. of semicolons	0.08
liwc.tentat	Freq. of tentative words	0.15	cm.DESWLsy	Mean word length	0.07
liwc.reward	Freq. of words associated with reward	0.14	liwc.adverb	Freq. of adverbs	0.07
liwc.informal	Freq. of informal words	0.14	liwc.social	Freq. of words related to social processes	0.07
cm.WRDPRP2	Freq. of second person pronouns	0.14	liwc.article	Freq. of articles	0.07
FS			**FT**		
Variable	**Description**	**SHAP**	**Variable**	**Description**	**SHAP**
cm.CRFNO1	Noun overlap between adjunct sentences	0.56	liwc.SemiC	Freq. of semicolons	0.39
cm.WRDPRP3s	Freq. of third person pronouns	0.50	cm.LSASS1	LSA measure of semantic coherence	0.19
cm.CRFSO1	Word stem overlap between adjunct sentences	0.43	cm.CNCNeg	Freq. of negative connectives	0.12
cm.DRAP	Adverbial phrase density	0.35	liwc.adverb	Freq. of adverbs	0.11
cm.CRFCWOa	Content word overlap of all sentences	0.25	cm.DESWLltd	Standard deviation of average no. of letters/word	0.09
liwc.risk	Freq. of risk related words	0.23	liwc.space	Freq. of words related to space	0.09
liwc.differ	Freq. of words related to differentiation	0.21	liwc.verb	Freq. of verbs	0.08
liwc.focusfuture	Freq. of future focus words	0.21	liwc.shehe	Freq. of third person singular pronouns	0.07
liwc.focuspresent	Freq. of present focus words	0.20	cm.SYNLE	Mean no. of words before the main verb	0.06
liwc.affiliation	Freq. of affiliation words	0.16	liwc.discrep	Freq. of words associated with discrepancy	0.06

Table 5: For RQ2 classifiers are exclusively trained on English (EN) and Portuguese (PT) feedback examples. Performance of each classifier is measured against EN and PT feedback examples. Legend: ACC – Accuracy; K – Cohen's kappa; F1 - F1 Score.

		FS			FT			FP		
		ACC	K	F1	ACC	K	F1	ACC	K	F1
EN Classifier	EN	0.83	**0.42**	**0.52**	**0.69**	**0.13**	**0.31**	**0.66**	**0.23**	**0.49**
	PT	**0.85**	0.03	0.04	0.11	0.00	0.00	0.49	-0.02	0.30
PT Classifier	EN	0.79	0.06	0.12	0.28	0.00	0.43	0.35	0.00	0.52
	PT	**0.94**	**0.74**	**0.77**	**0.91**	**0.49**	**0.95**	**0.78**	**0.56**	**0.79**

features across all levels of feedback.

One possible explanation for this finding might be the difference in courses represented in the English and Portuguese datasets. English feedback examples were primarily from STEM related courses, including Environmental Studies and Software Engineering, while Portuguese feedback examples had more of a mix, hailing from Biology and Literature courses. Hence, the different nature of represented courses might have influenced the transferability analysis.

Another explanation for the low transferability of features might be the cultural differences in communication. For instance, at the FS level of feedback, we observed greater association of friendship and social processes for the English feedback; i.e. English instructors might have displayed a greater level of familiarity with students. As an instructor can be viewed as an authority figure, this difference might be related to whether a culture is "horizontal", and therefore emphasizes equality, or "vertical", and emphasizes hierarchy [25]. The implications of this finding would indicate instructors will need to consider the cultural backgrounds of the learner while delivering feedback for improved efficacy.

4. CONCLUSION AND FUTURE RESEARCH

This study proposed four main contributions. First, this study explored how accurately a trained model can identify the presence of different feedback practices. The constructed classifiers, using primarily linguistic and psychological features, were effective in identifying the presence of FT, FP and FS levels of feedback and showed better performance than similar works in this content area; however, the FR classifier was marred by a lack of adequate data. The implications of these results provide a proof of concept for a tool that can automatically analyze and potentially diagnose the contents of an instructor's feedback. This promotes the understanding and utilization of good feedback practices to improve their efficacy on learner adoption.

Another goal of this paper was to identify the prominent textual features of good feedback practices. Identified features were able to corroborate the findings of educational research on feedback theory. The presented findings can be further used to inspire the design of future automated feedback generators, e.g., intentionally including the prominent terms specific to different feedback practices when generating feedback.

Finally, this study conducted an analysis of the transferability of feedback features across languages. Feedback tools should be generalizable enough to cater to a variety of languages. By analyzing the transferability of feedback features across languages, this study aimed to enhance the global adaptability of current and future feedback tools. The findings indicate feedback features have low transferability between feedback examples delivered in English and Portuguese. However, a more expansive study is suggested, with a greater size and variety of feedback from different languages.

References

[1] W. K. Balzer, M. E. Doherty, and R. O'Connor. Effects of cognitive feedback on performance. *Psychol. Bull.*, 106(3):410–433, 1989. ISSN 1939-1455, 0033-2909.

[2] G. Barbosa, R. Camelo, A. P. Cavalcanti, P. Miranda, R. F. Mello, V. Kovanović, and D. Gašević. Towards automatic cross-language classification of cognitive presence in online discussions. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, LAK '20, pages 605–614, Frankfurt, Germany, Mar. 2020. ACM.

[3] D. L. Butler and P. H. Winne. Feedback and Self-Regulated Learning: A Theoretical Synthesis. *Review of Educational Research*, 65(3):245–281, Sept. 1995. ISSN 0034-6543, 1935-1046.

[4] A. P. Cavalcanti, A. Diego, R. F. Mello, K. Mangaroska, A. Nascimento, F. Freitas, and D. Gašević. How good is my feedback?: a content analysis of written feedback. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 428–437, Frankfurt Germany, Mar. 2020. ACM.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Oversampling Technique. *JAIR*, 16:321–357, June 2002. ISSN 1076-9757.

[6] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, San Francisco, California, USA, Aug. 2016. ACM.

[7] J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20 (1):37–46, Apr. 1960. ISSN 0013-1644, 1552-3888.

[8] D. Denisko and M. M. Hoffman. Classification and interaction in random forests. *Proceedings of the National Academy of Sciences*, 115(8):1690–1692, Feb. 2018. ISSN 0027-8424, 1091-6490.

[9] P. Ferguson. Student perceptions of quality feedback in teacher education. *Assess Eval High Educ*, 36(1):51–62, Jan. 2011. ISSN 0260-2938.

[10] C. Glover and E. Brown. Written Feedback for Students: too much, too detailed or too incomprehensible to be effective? *Bioscience Education*, 7(1):1–16, May 2006. ISSN null.

[11] A. C. Graesser, D. S. McNamara, M. M. Louwerse, and Z. Cai. Coh-Metrix: Analysis of text on cohesion and language. *Behav Res Methods Instrum Comput.*, 36(2): 193–202, May 2004. ISSN 0743-3808, 1532-5970.

[12] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer series in statistics. Springer, New York, NY, 2nd ed edition, 2009.

[13] J. Hattie and M. Gan. Instruction based on feedback. *Handbook of research on learning and instruction*, pages 249–271, 2011.

[14] J. Hattie and H. Timperley. The Power of Feedback. *Review of Educational Research*, 77(1):81–112, Mar. 2007. ISSN 0034-6543.

[15] H. Keuning, J. Jeuring, and B. Heeren. A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education (TOCE)*, 19(1):1–43, 2018.

[16] V. Kovanović, S. Joksimović, Z. Waters, D. Gašević, K. Kitto, M. Hatala, and G. Siemens. Towards automated content analysis of discussion transcripts: a cognitive presence case. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, LAK '16, pages 15–24, Edinburgh, United Kingdom, Apr. 2016. ACM.

[17] M. Liu, Y. Li, W. Xu, and L. Liu. Automated Essay Feedback Generation and Its Impact on Revision. *IEEE Trans. Learn. Technol.*, 10(4):502–513, Oct. 2017. ISSN 1939-1382.

[18] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. Explainable AI for Trees: From Local Explanations to Global Understanding. May 2019.

[19] X. Ma, S. Wijewickrema, S. Zhou, Y. Zhou, Z. Mhammedi, S. O'Leary, and J. Bailey. Adversarial Generation of Real-time Feedback with Neural Networks for Simulation-based Training. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3763–3769, Melbourne, Australia, Aug. 2017. International Joint Conferences on Artificial Intelligence Organization.

[20] K. A. Neuendorf. *The content analysis guidebook.* SAGE, Los Angeles, second edition edition, 2017.

[21] D. J. Nicol and D. Macfarlane-Dick. Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. *Studies in Higher Education*, 31(2):199–218, Apr. 2006. ISSN 0307-5079, 1470-174X.

[22] B. Pan. Application of XGBoost algorithm in hourly PM2.5 concentration prediction. *IOP Conference Series: Earth and Environmental Science*, 113:012127, Feb. 2018. ISSN 1755-1315. Publisher: IOP Publishing.

[23] A. Parikh, K. McReelis, and B. Hodges. Student feedback in problem based learning: a survey of 103 final year students across five Ontario medical schools. *Med. Educ.*, 35(7):632–636, 2001. ISSN 1365-2923.

[24] A. Pinheiro Cavalcanti, R. Ferreira Leite de Mello, V. Rolim, M. Andre, F. Freitas, and D. Gasevic. An Analysis of the use of Good Feedback Practices in Online Learning Courses. In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, pages 153–157, Maceió, Brazil, July 2019. IEEE.

[25] S. Shavitt, A. K. Lalwani, J. Zhang, and C. J. Torelli. The Horizontal/Vertical Distinction in Cross-Cultural Consumer Research. *Journal of Consumer Psychology*, 16(4):325–342, 2006. ISSN 1532-7663. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15327663jcp1604_3

[26] P.-N. Tan and others. *Introduction to data mining*. Pearson Education India, 2007.

[27] Y. R. Tausczik and J. W. Pennebaker. The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods. *JLS*, 29(1):24–54, Mar. 2010. ISSN 0261-927X, 1552-6526.

[28] J. Villalón, P. Kearney, R. A. Calvo, and P. Reimann. Glosser: Enhanced Feedback for Student Writing Tasks. In *2008 Eighth IEEE International Conference on Advanced Learning Technologies*, pages 454–458, Santander, Cantabria, Spain, 2008. IEEE.

[29] M. R. Weaver. Do students value feedback? Student perceptions of tutors' written responses. *Assess Eval High Educ*, 31(3):379–394, June 2006. ISSN 0260-2938.

[30] S. Wijewickrema, X. Ma, P. Piromchai, R. Briggs, J. Bailey, G. Kennedy, and S. O'Leary. Providing Automated Real-Time Technical Feedback for Virtual Reality Based Surgical Training: Is the Simpler the Better? In *Artificial Intelligence in Education*, Lecture Notes in Computer Science, pages 584–598, Cham, 2018. Springer.

[31] Z. Xiao, Y. Wang, K. Fu, and F. Wu. Identifying Different Transportation Modes from Trajectory Data Using Tree-Based Ensemble Classifiers. *ISPRS International Journal of Geo-Information*, 6(2):57, Feb. 2017. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

Academic Integrity during the COVID-19 Pandemic: a Social Media Mining Study

Mohammad S. Parsa
University of Waterloo
mohammad.parsa@uwaterloo.ca

Lukasz Golab
University of Waterloo
lgolab@uwaterloo.ca

ABSTRACT

Academic integrity has been a frequently reported challenge in online education. Given the widespread transition to online program delivery during the COVID-19 pandemic, we ask the following question: *How do college students feel about online cheating?* Our analysis is based on academic discussions on the Reddit social curation platform in Fall 2020 and, for comparison, Fall 2019. We found more discussions related to cheating in 2020 than in 2019, and the topics have expanded from plagiarism in programming assignments to online assessments in general. Topic modelling of the Fall 2020 discussions revealed three concerns raised by students: that cheating inflates grades and forces instructors to increase the difficulty of assessments; that witnessing cheating go unpunished is demotivating; and that academic integrity policies are not always communicated clearly.

Keywords

academic integrity, online education, social media, text mining

1. INTRODUCTION

Recent studies have reported that online academic misconduct has increased during the COVID-19 pandemic [12, 6, 2, 4, 3, 18]. We therefore ask the following question in this paper: *How do college students feel about online cheating?* To answer this question, we turn the Reddit social curation platform (reddit.com). Reddit hosts over 100,000 user-created discussion communities refereed to as *subreddits*. Within a subreddit, users create posts that other users comment on. Subreddit names begin with "r/" and correspond to the subreddit topic, e.g., r/politics or r/relationship_advice.

Descriptive subreddit names make it easy to locate discussions about specific topics or discussions initiated by various kinds of users. Of interest to our study are over 80 subreddits corresponding to Canadian and U.S. universities, which we call *academic subreddits*. We collected all posts and comments on academic subreddits created during the Fall 2019 and Fall 2020 semesters (September through December inclusive) that match at least one keyword related to cheating, such as 'cheat' or 'misconduct'.

Mohammad Parsa and Lukasz Golab "Academic Integrity during the COVID-19 Pandemic: a Social Media Mining Study". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 777-781. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Our analysis consists of two steps. First, collecting data from the same time period in 2019 and 2020 allows us to compare cheating-oriented discussions from before the pandemic, when classes were held in person, and during the pandemic, with most courses delivered online. To do so, we train a logistic regression classifier to distinguish between Fall 2019 and Fall 2020 content based on the words used. Next, we analyze Fall 2020 discussions in detail. We apply the Non-negative Matrix Factorization algorithm [20], which clusters posts and comments based on the words used and allows us to identify common discussion topics.

Related Work: Social media have become a go-to source of public opinion on a variety of topics. In particular, academic subreddits have been analyzed in recent work on students' mental health [1, 16], but academic integrity was not discussed. The closest works to ours are those in [4] and [5], which interviewed a small set of undergraduate students and educators. The participants identified some positive aspects of online education, but expressed concerns about cheating and the level of difficulty of online assessments. Our social media analysis explores these and other concerns in detail.

2. DATA AND METHODS

Previous work on students' mental health [1, 16] identified 83 *academic* subreddits corresponding to major U.S. and Canadian universities. We analyze the same subreddits in this paper, listed in the first column of Table 1 (U.S.) and Table 2 (Canadian). We collected all posts and comments on these subreddits from the Fall 2019 semester, when classes and examinations were held in person, and the Fall 2020 semester, when most campuses moved to online delivery (September-December inclusive). We downloaded the data using a publicly-accessible Reddit interface at pushshift.io.

Next, we retain only those posts and comments that contain at least one of the following keywords: 'cheat', 'plagiari', and 'misconduct'. We perform *substring* matching, meaning that 'plagari' also matches 'plagiarize' and 'plagiarism'. Tables 1 and 2 report the number of posts ("P") and comments ("C") on each U.S. and Canadian academic subreddit, respectively, in Fall 2019 and Fall 2020. The "Before" numbers correspond to all posts and comments. The "After" numbers correspond to posts and comments that matched at least one cheating-related keyword; note that there are three times as many such posts and comments in 2020 than in 2019 (7,809 vs. 2,524) even though the total number of posts and comments on academic subreddits has not changed much from 2019 to 2020 (see the total "Before" numbers in the last row of Tables 1 and 2).

We then perform standard text pre-processing. Following previous work on Reddit topic modelling [10, 16], we remove posts and

Table 1: Number of posts and comments on U.S. academic subreddits in 2019 and 2020 before and after filtering to find cheating-related discussions (C: Comments, P: Posts).

Subreddits	2020				2019			
	Before		After		Before		After	
	C	P	C	P	C	P	C	P
UIUC	39974	6991	160	21	40556	6431	104	10
berkeley	37355	6343	365	69	28537	4637	114	17
Cornell	36235	8139	165	27	22562	3900	45	8
Purdue	34376	6317	148	15	33322	5273	42	11
UCSD	30589	5798	175	34	28214	5364	106	15
rutgers	29861	6622	269	69	44114	8902	122	16
UMD	21937	4225	206	28	25794	4631	97	6
SBU	20521	4301	163	20	28328	5373	63	13
uofm	19954	3174	79	14	13553	2213	44	5
udub	17867	3487	82	17	18187	3187	59	6
UWMadison	14870	2447	103	18	14236	2039	33	3
UTAustin	13620	3112	53	7	13866	2811	90	6
utdallas	12763	2235	74	7	20731	3109	25	5
PennStateUniversity	12345	1944	64	5	9620	1610	42	2
msu	12052	2104	86	10	15066	2329	23	6
NCSU	11653	1794	72	5	18943	2524	32	1
UVA	11627	2424	79	9	5071	1084	19	5
rit	11603	1577	48	2	10768	1643	6	1
nyu	11034	2952	37	7	5731	1438	10	2
UNCCharlotte	10132	1709	93	12	10700	1508	18	1
USC	9551	1958	82	15	6800	1419	17	4
Baruch	9370	2226	94	16	4851	1144	36	12
UPenn	8886	2083	55	10	4212	997	11	1
UNC	8347	1644	30	8	3800	790	6	2
byu	6951	707	39	2	3165	407	25	3
UGA	6637	1520	20	3	6852	1349	2	0
columbia	6496	1573	55	5	4699	708	22	3
RPI	5652	1220	70	0	7622	1343	5	0
uichicago	4880	894	46	4	6606	1009	84	1
SJSU	4661	1068	27	5	5108	1136	18	3
stanford	3944	1223	13	2	3782	882	10	0
bostoncollege	3493	1006	0	0	753	188	0	0
cmu	3388	657	27	2	2764	517	3	0
washu	3159	572	4	0	1134	259	0	0
Vanderbilt	2581	555	9	1	1447	311	0	0
Harvard	2219	634	1	1	2294	517	1	0
UMBC	2036	457	21	3	2479	464	4	0
duke	2020	469	2	1	1397	317	7	2
mit	1758	532	3	0	1651	373	4	0
BrownU	1363	438	2	1	1315	276	0	0
IndianaUniversity	1225	588	1	1	1797	543	9	1
Caltech	494	130	0	0	220	59	0	0
Total	509479	99849	3122	476	482647	85014	1358	171

comments with fewer than 40 or more than 4000 characters: short ones are unlikely to be meaningful (and may correspond to URLs), while long ones may mention more than one topic. We also remove stopwords and lemmatize the remaining words using the Python NLTK parser.

To distinguish between cheating-related discussions before and during the pandemic, we train a logistic regression classifier to predict whether a post or comment was written in Fall 2020 or Fall 2019. We use term frequency–inverse document frequency (TF-IDF) word scores as features in the model. We chose logistic regression due to its interpretable nature: words with positive coefficients represent Fall 2020 content and words with negative coefficients represent Fall 2019. Our model obtained a 10-fold cross-validation accuracy score of 73%, a precision of 76%, a recall of 96% and an F1-score of 86%.

(We also tested logistic regression models with additional features, including word bigrams, the sentiment of the post or comment (computed using the Valence Aware Dictionary and Sentiment Reasoner (VADER) [8]) and linguistic features computed using Linguistic Inquiry and Word Count (LIWC) [17]. After adding these features, accuracy improved by two percent to 75%. However, none of these additional features were assigned large coefficients and therefore are not considered further in the remainder of the paper.)

Finally, we apply the Non-negative Matrix Factorization (NMF) topic modelling algorithm [20], which was used in prior work on Reddit mining [14, 7, 11], on the Fall 2020 posts and comments that match at least one cheating-related keyword. We again represent each post and comment using the TF-IDF scores of the words occurring in it. NMF clusters documents into topics and assigns a list of representative terms called *topic descriptors* to each topic. NMF also calculates the "representativeness" score of each topic descriptor, and we report the top-10 highest-scoring descriptors for each topic. Moreover, we report top-10 frequent word n-grams (for n up to three, i.e., sequences of up to three consecutive words) for each topic.

Table 2: Number of posts and comments on Canadian academic subreddits in 2019 and 2020 before and after filtering to find cheating-related discussions (C: Comments, P: Posts).

Subreddits	2020				2019			
	Before		After		Before		After	
	C	P	C	P	C	P	C	P
uwaterloo	72244	8372	381	58	88996	9888	130	17
UofT	54343	8460	701	86	67649	9375	171	23
UBC	40058	5281	766	42	39416	5039	109	11
uAlberta	33265	7164	341	58	49494	8270	137	23
McMaster	24556	5188	219	45	14932	2638	27	3
mcgill	21380	3376	167	15	20852	3067	58	6
yorku	15671	4065	228	46	22078	3862	47	6
CarletonU	15455	2531	207	11	16874	2706	43	2
Concordia	10065	2394	192	27	10292	2185	27	7
uwo	9717	1856	122	10	11758	1764	35	2
wlu	8097	1788	97	16	5499	1203	13	4
uvic	7291	1178	85	3	4756	828	11	3
ryerson	6503	2282	87	6	14922	2927	37	8
queensuniversity	5234	1107	18	1	4758	824	6	2
umanitoba	4408	861	66	7	3183	717	3	1
uoguelph	3381	794	51	8	3691	693	5	2
Dalhousie	1807	401	21	4	2019	407	6	2
usask	1177	290	0	0	666	178	0	0
brocku	1007	366	2	0	1442	329	4	2
memorialuniversity	785	183	6	1	637	147	2	0
UdeM	422	90	1	0	174	48	0	0
lakeheadu	119	59	2	1	51	21	0	0
uleth	112	35	0	0	82	33	0	0
University_Of_Regina	96	30	1	0	8	11	0	0
AcadiaU	69	29	1	0	60	15	0	0
UQAM	67	22	0	0	48	17	0	0
uwinnipeg	65	24	2	1	15	10	0	0
unb	62	35	0	1	8	12	0	0
laurentian	33	16	0	0	9	4	0	0
stfx	32	12	0	0	0	1	0	0
SMUHalifax	24	17	0	0	21	9	0	0
nipissingu	13	8	0	0	3	4	0	0
UPEI	12	10	0	0	1	3	0	0
stthomas	6	4	0	0	0	3	0	0
BishopUniversity	5	2	0	0	0	4	0	0
UNBC	3	5	0	0	15	10	0	0
mta	1	0	0	0	6	6	0	0
cbu	0	2	0	0	3	1	0	0
MSVU	0	0	0	0	0	1	0	0
uottawa	0	0	0	0	83	43	0	0
usherbrooke	0	0	0	0	0	2	0	0
Total	337585	58337	3764	447	384501	57305	871	124

Additionally, NMF assigns a *closeness score* for each document-topic pair, indicating how close the document is to a topic. To obtain more information about the topics produced by NMF, for each topic, we manually inspect 5% of the posts and comments with the highest closeness scores.

NMF requires the number of topics as input. Following previous work [15], we run NMF to produce between 5 and 50 topics and compute the *coherence* score for each. Coherence measures the extent to which the top representative terms representing each topic are semantically related (higher is better). We obtained the highest scores for 5 and 20 topics. A preliminary analysis of the NMF output at five topics revealed that most topics consisted of several discussion themes. This observation suggested that a larger number of topics may be more appropriate, and thus we selected 20 topics.

3. RESULTS

We begin with the results of our logistic regression analysis, shown in Table 4 in the Appendix. The most positive coefficients, predicting Fall 2020 posts and comments, include 'chegg' (an online platform for answering college and high school questions), as well as words related to online proctoring such as 'proctor', 'proctorio', 'zoom', 'camera', 'webcam' and 'privacy'. The most negative coefficients, predicting Fall 2019 posts and comments, suggest in-person examinations ('cheat sheet', 'bring', 'sit') and programming assignments and projects ('code', 'program', 'project').

Next, we move to topic modelling. Table 3 shows the NMF topic descriptors, the frequent n-grams, and the percentage of posts and comments assigned to each topic. We group the topics into the following three categories based on the information in Table 3 and manual inspection of a sample of posts and comments.

First, about 40% of the posts and comments include concerns about cheating leading to grade inflation, which in turn leads to assessments becoming more difficult. Students have observed grade inflation (Topic 13) and expressed concerns that Fall 2020 examinations will be more difficult to reduce the class average (Topics 1 and 20). Moreover, students commented on various methods used by

Table 3: Fall 2020 topic modelling results

#	Topic descriptors	Frequent N-grams	%
1	work, really, time, way, learn, try, hard, help, school, good	*'feel like', 'work hard', 'first year', 'high school', 'office hour', 'mental health', 'learn material', 'get catch', 'make sure', 'in person'*	10.4
2	say, academic, email, integrity, case, code, worry, report, flag, mean	*'academic integrity', 'academic dishonesty', 'integrity violation', 'academic integrity violation', 'get flag', 'student conduct', 'academic offense', 'would say', 'get catch', 'even though'*	10.3
3	think, probably, pretty, fine, worry, fair, sure, reason, away, good	*'think would', 'think people', 'think get', 'like think', 'get away', 'make sure', 'really think', 'feel like', 'think go', 'think make'*	6.4
4	student, university, honest, case, punish, international, chinese, issue, school, conduct	*'international student', 'student get', 'many student', 'chinese student', 'honest student', 'academic integrity', 'student would', 'mental health', 'academic dishonesty', 'first year'*	5.7
5	know, want, let, wrong, happen, person, tell, need, mean, consequence	*'let know', 'want know', 'know people', 'get catch', 'know would', 'lot people', 'feel like', 'know know', 'know go', 'student know'*	5.5
6	prof, email, mark, ta, ask, tell, send, chance, midterm, try	*'prof make', 'first year', 'email prof', 'open book', 'feel like', 'prof say', 'prof ta', 'prof would', 'make sure', 'ask prof'*	5.4
7	question, answer, time, ask, quiz, look, minute, similar, wrong, google	*'answer question', 'go back', 'multiple choice', 'short answer', 'exam question', 'one question', 'look answer', 'question answer', 'question exam', 'choice question'*	5.1
8	test, open, book, note, close, online, tab, internet, easy, search	*'open book', 'open note', 'make test', 'take test', 'test open', 'close book', 'book exam', 'open book exam', 'exam open', 'book test'*	4.9
9	people, lot, stop, say, agree, mean, proctor, probably, maybe, care	*'people get', 'lot people', 'many people', 'people would', 'get catch', 'people like', 'mental health', 'people go', 'know people', 'feel like'*	4.8
10	like, feel, sound, look, yeah, lol, bad, thing, lot, shit	*'feel like', 'seem like', 'look like', 'sound like', 'something like', 'even though', 'would like', 'make feel', 'online school', 'like people'*	4.8
11	exam, proctor, final, online, open, book, sheet, time, hour, note	*'take exam', 'final exam', 'open book', 'online exam', 'make exam', 'proctor exam', 'write exam', 'take home', 'home exam', 'person exam'*	4.7
12	use, software, proctor, proctorio, computer, browser, note, flag, lockdown, webcam	*'lockdown browser', 'secondary device', 'make sure', 'proctor software', 'take exam', 'get flag', 'student use', 'use respondus', 'virtual machine', 'use note'*	4.5
13	course, year, average, math, midterm, final, assignment, fail, term, quiz	*'first year', 'take course', 'last year', 'math course', 'feel like', 'midterm final', 'year course', 'course average', 'final exam', 'class average'*	4.5
14	class, curve, online, semester, average, fail, homework, lot, easy, problem	*'take class', 'class average', 'online class', 'class get', 'one class', 'feel like', 'math class', 'class take', 'in person', 'make sure'*	4.4
15	grade, curve, average, semester, high, final, letter, higher, better, good	*'good grade', 'letter grade', 'final grade', 'get good', 'get good grade', 'grade get', 'get grade', 'grade inflation', 'grade curve', 'better grade'*	4.2
16	professor, happen, try, evidence, accuse, report, tell, prove, probably, email	*'professor make', 'take exam', 'make exam', 'professor would', 'professor might', 'make sure', 'student professor', 'professor try', 'in person', 'tell professor'*	4
17	catch, happen, wonder, lol, hear, dumb, expel, time, lmao, guy	*'get catch', 'people get', 'people get catch', 'first time', 'catch people', 'catch get', 'use chegg', 'get away', 'without get', 'without get catch'*	3.7
18	chegg, post, account, use, ip, information, address, answer, view, solution	*'use chegg', 'ip address', 'chegg account', 'get catch', 'post chegg', 'question chegg', 'post question', 'chegg exam', 'chegg answer', 'answer chegg*	2.8
19	group, chat, leave, join, share, report, quiz, snitch, post, want	*'group chat', 'share answer', 'get trouble', 'group member', 'join group', 'class group', 'leave group', 'academic integrity', 'group project', 'study group'*	2.5
20	make, harder, sure, sense, hard, easier, difficult, mistake, thing, pretty	*'make sure', 'make harder', 'make exam', 'make sense', 'harder make', 'want make', 'make mistake', 'make difficult', 'make feel', 'want make sure*	1.4

instructors to combat cheating and reduce grades, such as grading on a curve (Topics 14 and 15) and using anti-cheating and online proctoring software (Topics 9 and 11).

Next, students reported feeling demotivated when they know that cheating happens in examinations (Topics 4 and 5) and often goes unpunished (Topics 3, 10 and 17). Students discussed examples of cheating that instructors failed to identify, such as seeking answers on Google and question-answering websites such as Chegg (Topics 7, 8 and 18), and discussing solutions in online chat groups (Topic 19).

Finally, students reported concerns about new methods used to prevent cheating in online examinations. They worried that some legitimate actions may be misconstrued as cheating: looking away from the computer screen, accidentally pressing a button, or disconnecting from a video meeting due to internet connectivity issues (Topics 6 and 12). Furthermore, some students reported being accused of cheating during online examinations, but did not realize they did anything wrong (Topics 2 and 16).

4. CONCLUSIONS

Logistic regression analysis suggests that cheating-related discussions on academic subreddits have expanded from plagiarism in computer programming (representative of Fall 2019) to online assessments in general. The word 'chegg' was associated with Fall 2020 content, suggesting an increase in the use of Chegg and related websites, which is consistent with prior work [6, 3]. Furthermore, words indicating online proctoring were predictive of Fall 2020 content, e.g., 'camera', 'webcam' and 'record'. Inspection of the posts and comments containing these terms revealed students' concerns about their privacy during online examinations. Similar concerns were raised in recent work [4, 9].

Topic modelling analysis identified three discussion themes in Fall 2020. First, students believe that cheating causes grade inflation, which motivates instructors to make assessments harder and introduce strict anti-cheating protocols such as not being able to scroll back to a previous question on an online examination. Some of these concerns have been highlighted in previous work [18, 19, 2, 4, 13, 3], and our analysis reflects students' opinions on this topic. Second, unpunished cheating lowers students' morale and motivation. Students report feeling demotivated when classmates cheat and obtain high grades. Third, students report not knowing exactly what constitutes cheating and what is allowed, underscoring the im-

portance of clear academic integrity policies. These concerns were often reported in the context of online examinations, with students unsure of how their actions are being monitored.

5. REFERENCES

[1] S. Bagroy, P. Kumaraguru, and M. D. Choudhury. A social media based index of mental well-being in college campuses. In *CHI*, pages 1634–1646, 2017.

[2] E. Bilen and A. Matros. Online cheating amid COVID-19. *Journal of Economic Behavior & Organization*, 182:196–211, Feb. 2021.

[3] T. M. Clark, C. S. Callam, N. M. Paul, M. W. Stoltzfus, and D. Turner. Testing in the time of COVID-19: A sudden transition to unproctored online exams. *Journal of Chemical Education*, 97(9):3413–3417, July 2020.

[4] J. R. Deters, M. C. Paretti, and J. M. Case. How implicit assumptions about engineering impacted teaching and learning during COVID-19. *Advances in Engineering Education*, 8(4):1–5, 2020.

[5] B. Dorić, M. Blagojević, M. Papic, and N. Stanković. Students' attitudes regarding online learning during COVID-19 pandemic. In *Information Technology and Education Development*, pages 157–160, 2020.

[6] K. A. Gamage, E. K. de Silva, and N. Gunawardhana. Online delivery and assessment during COVID-19: Safeguarding academic integrity. *Education Sciences*, 10(11):301, Oct. 2020.

[7] N. Gozzi, M. Tizzani, M. Starnini, F. Ciulla, D. Paolotti, A. Panisson, and N. Perra. Collective response to media coverage of the COVID-19 pandemic on reddit and wikipedia: Mixed-methods analysis. *Journal of Medical Internet Research*, 22(10):e21597, Oct. 2020.

[8] C. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*, pages 216–225, 2015.

[9] M. V. Jamieson. Keeping a learning community and academic integrity intact after a mid-term shift to online learning in chemical engineering design during the COVID-19 pandemic. *Journal of Chemical Education*, 97(9):2768–2772, Aug. 2020.

[10] A. Khan and L. Golab. Reddit mining to understand gendered movements. In *Proc. EDBT Workshop on Data Analytics Solutions for Real-Life Applications*, pages 3:1–3:8, 2020.

[11] H. Liu, Q. Li, R. Yao, and D. D. Zeng. Analyzing topics of JUUL discussions on social media using a semantics-assisted NMF model. In *ISI*, pages 212–214, 2019.

[12] D. M. Low, L. Rumker, T. Talkar, J. Torous, G. Cecchi, and S. S. Ghosh. Natural language processing reveals vulnerable mental health support groups and heightened health anxiety on reddit during COVID-19: Observational study. *Journal of Medical Internet Research*, 22(10):e22635, Oct. 2020.

[13] C. K. C. Ng. Evaluation of academic integrity of online open book assessments implemented in an undergraduate medical radiation science course during COVID-19 pandemic. *Journal of Medical Imaging and Radiation Sciences*, 51(4):610–616, Dec. 2020.

[14] A. Nobles, C. Dreisbach, J. Keim-Malpass, and L. Barnes. Is this an STD? please help!: Online information seeking for sexually transmitted diseases on reddit. *ICWSM*, pages 660–663, 2018.

[15] D. O'callaghan, D. Greene, J. Carthy, and P. Cunningham. An analysis of the coherence of descriptors in topic modeling. *Expert Systems with Applications*, 42(13):5645–5657, 2015.

[16] M. Parsa and L. Golab. Social media mining to understand the impact of co-operative education on mental health. In *EDM*, pages 653–657, 2020.

[17] J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn. The development and psychometric properties of LIWC 2015. Technical report, 2015.

[18] D. M. Telles-Langdon. Transitioning university courses online in response to COVID-19. *Journal of Teaching and Learning*, 14(1):108–119, May 2020.

[19] N. A. A. Tuah. Is online assessment in higher education institutions during COVID-19 pandemic reliable? *Siriraj Medical Journal*, 73(1):61–68, Dec. 2020.

[20] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *SIGIR*, pages 267–273, 2003.

APPENDIX

Table 4: Words with the most positive and most negative logistic regression coefficients

Term	coefficient	Term	coefficient
chegg	2.19	sheet	-3
online	1.79	cheat sheet	-2.95
proctor	1.79	code	-1.87
open	1.62	project	-1.68
covid	1.55	plagiarism	-1.51
zoom	1.45	phone	-1.47
prof	1.37	plagiarize	-1.32
pandemic	1.25	relationship	-1.31
proctorio	1.11	sit	-1.1
flag	1.09	talk	-1.02
cheat	1.08	sexual	-0.98
chat	1.06	notice	-0.94
camera	1.03	bring	-0.93
internet	1	textbook	-0.93
privacy	1	international	-0.92
book	1	misconduct	-0.78
cheater	0.98	appeal	-0.78
webcam	0.95	program	-0.79
100	0.93	go	-0.79
format	0.92	front	-0.81
screen	0.9	report	-0.81
open book	0.89	try cheat	-0.81
sem	0.88	ask	-0.81
record	0.88	homework	-0.82
math	0.88	dean	-0.82
term	0.87	practice	-0.83
average	0.86	allow	-0.88
respondus	0.85	partner	-0.88
email	0.83	final	-0.89
semester	0.83	english	-0.9

Analyzing Ranking Strategies to Characterize Competition for Co-Operative Work Placements

Shivangi Chopra
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
s9chopra@uwaterloo.ca

Lukasz Golab
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
lgolab@uwaterloo.ca

ABSTRACT

Co-operative education is a form of work-integrated learning that includes academic study and paid work experience. This provides new learning opportunities for students and a talent pipeline for employers, but also requires participation in a competitive job market. We study competition through a unique dataset from a large North American co-operative program, in which students and employers rank each other after a round of interviews, then a matching algorithm assigns students to jobs based on the ranks, and finally students and employers evaluate each other at the end of the workterm. Our results reveal insights about competition and its impact on decision-making and satisfaction. An analysis of common ranking patterns suggests that small employers appear to be more strongly affected by competition and consider more options in their rankings, whereas large employers often do not provide any backup options and only identify their top choice. Additionally, competition appears to affect satisfaction since employers give higher workterm evaluations when matched with their top choice.

Keywords

co-operative education, work-integrated learning, ranking

1. INTRODUCTION

Co-operative (co-op) education is a form of work-integrated learning that includes both academic study terms and paid work experience, referred to as co-op work placements, workterms or internships. Prior work has examined the benefits of co-op, such as new learning opportunities for students and a talent pipeline for employers [13]. However, recent work has also reported that the competition related to interviewing for and securing co-op placements is a source of stress for students [10]. Motivated by these findings, in this paper we take a closer look at competition in co-operative education.

Our study is based on a unique dataset from a large North American undergraduate co-operative program. In this program, the co-op employment process proceeds as follows. Employers post job advertisements, students submit applications, and employers select students they wish to interview. After a round of interviews, students and employers rank each other. A *matching* algorithm then assigns students to jobs based on the ranks, with the goal of minimizing the sum of the student and employer ranks. For example, if the employer offering job A ranks student B one and vice versa, then the algorithm is guaranteed to assign job A to student B. In some cases, however, students and employers may be matched with their second or third choices, or not be matched at all. Finally students and employers evaluate each other at the end of the workterm.

One way to characterize competition in such a process is to identify job postings that receive the most applications. However, even entry-level or less desirable job postings may receive many applications, mainly from junior students. Instead, we turn to the ranking step of the process as a novel way to characterize competition. We investigate the following questions:

1. Do employers use different ranking strategies that reflect the level of competition they face? For example, an employer who is confident in their ability to attract top students may rank their preferred student one and not rank any other students as backup options. On the other hand, a less confident employer may rank multiple students.
2. Does competition appear to affect satisfaction? Are employers happier if they are matched with their top-ranked choices?

To answer these questions, we analyze ranking and workterm evaluation data from over 4,500 employers participating in the job matching process in three semesters, from September 2015 to August 2016. We answer the first question by mining frequent ranking patterns and identifying representative attributes of employers that use these patterns. To answer the second question, we compare the average employer evaluation scores when matched with their first choice versus a backup choice.

Related Work: Labour market competition has been studied from several angles, including improving talent recruitment by recommending resumes to job postings [11, 16], and reducing turnover by assessing personnel fit when making hiring decisions [2, 6, 3]. Further, it was found that job seekers' perceptions of hiring success, informed by their past job search success and prior knowledge of the company, motivate their decision to apply for a job and affect their decision to accept a job offer [1, 12]. In co-operative education, there has been work on student and employer satistfaction [8, 5, 4], as well as on clustering job opportunities, suggesting that junior students compete with each other for entry-level jobs and

Shivangi Chopra and Lukasz Golab "Analyzing Ranking Strategies to Characterize Competition for Co-Operative Work Placements". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 782-786. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

senior students compete with each other for more advanced positions [7, 14]. To the best of our knowledge, this is the first work to characterize competition based on employer rankings in a co-op process. Access to this unique data allows us to draw new insights into competition in co-operative education that can help manage students' and employers' expectations and improve their satisfaction.

2. DATA AND METHODS

2.1 Co-operative Process Overview

We begin with an overview of the co-op process at the institution studied in this paper. Initially, participating employers submit job descriptions, and any student (enrolled in a co-op program) may apply to any job. Next, employers interview selected candidates and rank them. A rank of zero, referred to as a "No Rank", means that the employer is not willing to hire the student. A rank of one, referred to as an "Offer", indicates that the employer wishes to hire the student. Ranks two to nine, referred to as "Ranks", represent the employer's backup or shortlist options, in order of preference. In other words, the employer would consider hiring these students if the top-ranked student declines the offer. In the remainder of this paper, we use the terms "shortlisted" and "received a Rank" interchangeably. Ranks do not need to be distinct, e.g., an employer may put five students on the backup list and give all of them a rank of two. After employers have submitted their rankings, students rank employers that made them offers or shortlisted them, between one and nine, indicating their order of preference.

The co-op matching system then removes student-employer rank pairs that add to zero (i.e., No Ranks) and applies a matching algorithm to assign students to jobs. The objective of the algorithm is to minimize the sum of the ranks of the resulting student-job assignment. Note that the lowest sum of ranks is two, and occurs when an employer offers a job to a student and the student gives a rank of one to this job. In this case, the student is guaranteed to be matched with this job[1]. In other cases, students or employers may be matched with their second, third, or lower choice, or may not be matched at all. Finally, at the end of a workterm, students and employers who were matched with each other evaluate each other.

2.2 Data

We analyzed one year of data, from September 2015 to August 2016, corresponding to 4,851 co-op job postings for students enrolled in co-op engineering programs:

- **Job Postings**, containing a job ID, job title, and employer name.
- **Employer Rankings**, containing a job ID and the distribution of ranks. Figure 1 shows an example with five employers, one per row. The first row indicates that employer (whose job ID is) E1 gave two ranks of zero (#R0) and no other ranks, i.e., E1 interviewed two students and was not willing to hire either of them. The second row indicates that E2 interviewed two students, rejected one (#R0), and put one on the shortlist with a rank of two (#R2), and so on.
- **Employer Evaluations**, containing a job ID, the rank the employer gave to the student who was hired, and the employer's evaluation of the student (on a 7-point scale: unsatisfactory, marginal, satisfactory, good, very good, excellent, outstanding).

[1]If a student were to give a rank of one to multiple Offers, the algorithm would randomly select one of these Offers.

	#R0	#R1	#R2	#R3	#R4	#R5	#R6	#R7	#R8	#R9
E1	2	0	0	0	0	0	0	0	0	0
E2	1	0	1	0	0	0	0	0	0	0
E3	4	2	0	0	0	0	0	0	0	0
E4	3	1	1	0	0	0	0	0	0	0
E5	1	1	1	1	1	1	0	0	0	0

Figure 1: Sample of employer ranking data

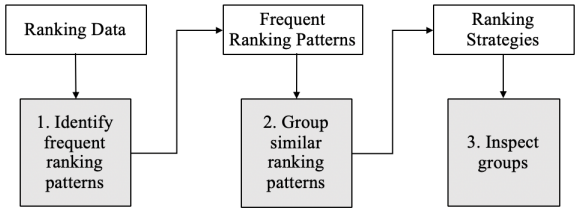


Figure 2: Summary of methods

2.3 Methods

Given that the matching algorithm is designed to minimize the sum of the ranks of the student-job assignments, employers may use different ranking strategies depending on the perceived level of competition. For example, employers may extend one or more offers but not shortlist any students if they are confident that their offer(s) will be accepted (i.e., that those students will reciprocate with a student rank of one). On the other hand, less confident employers may shortlist multiple students, and, to maximize their chances of hiring someone, they may give a rank of two to all shortlisted students instead of ranking them in order of preference.

The goal of this paper is to identify these kinds of ranking strategies and use them to describe the level of competition faced by different groups of employers. Our methodology, consisting of three steps, is summarized in Figure 2 and explained below.

1. **Identify frequent ranking patterns:** For employers, we identify commonly used *sets* of ranks. For example, an employer set of ranks of {0, 1} corresponds to employers who give only No Ranks (0) and Offers (1), and do not shortlist any students (ranks 2-9).
2. **Group similar ranking patterns:** Informed by the previous step and by the nature of the matching process, we group together similar sets of ranks. We refer to these as ranking strategies. For example, we may group employer rank sets of {0,1,2}, {0,1,2,3} and so on and label these as employers who make a shortlist (in addition to making some offers and rejecting some students). This step partitions employers according to their ranking strategies.
3. **Inspect groups:** We compare groups of employers with different ranking strategies based on their a) characteristics and, b) consequences on matching and evaluation. To identify differences among employers who use different ranking strategies, we inspect employer names and job titles. To understand the consequences of ranking strategies on matching

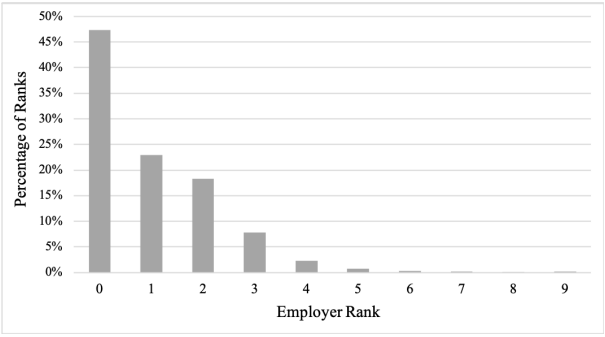


Figure 3: Distribution of employer ranks

Table 1: Most frequent sets of ranks given by employers

Set of Ranks	%
{0, 1, 2}	24
{0, 1}	19
{0, 1, 2, 3}	14
{1}	8
{0}	5
{1, 2, 3}	4
{1, 2}	4
{0, 1, 2, 3, 4}	4
{0, 2}	2
{1, 2, 3, 4}	2

and evaluation, for each group of employers who use a given ranking strategy, we calculate, a) the percentage who were not matched (represented as %NoMatch), b) the percentage who were matched with their first choice (represented as %MatchR1), and c) the percentage who were matched with their >1 choice (represented as %MatchR>1). Finally, we report the average evaluation scores that the employers gave to the students they matched with at the end of the workterm.

3. RESULTS

Section 3.1 analyzes the rankings given by 4,851 employers to identify frequent ranking patterns (Step 1 of Figure 2), group them into ranking strategies (Step 2 of Figure 2), and distinguish between employers with different ranking strategies (Step 3 of Figure 2). Section 3.2 analyzes the effects of ranking strategies on matching and satisfaction (Step 3 of Figure 2).

3.1 Employer Ranking Strategies

Figure 3 shows the distribution of ranks given by employers to students they have interviewed. Recall that rank 0 or "No Rank" indicates that the student was interviewed but not considered for the job, rank 1 represents an offer, and ranks 2-9 represent employers' shortlists in order of preference. As seen in Figure 3, nearly half the ranks are zero, a quarter are offers, and ranks lower than three are rare.

Next, Table 1 shows the most frequent *sets* of ranks given by employers. Many employers reject at least one student (rank 0), make at least one offer (rank 1), and shortlist at least one student, usually with ranks of 2 and/or 3. 19% of employers make offers without shortlisting anyone (second row: {0,1}).

Using Figure 3 and Table 1, we group employers with similar ranking patterns (Step 2 of Figure 2). Table 2 summarizes the groups. The first column, Label, describes each group. For example, the first group corresponds to employers that do not make any offers and do not shortlist (Rank) any students – that is, they only give zero ranks, meaning that they are not willing to hire any students they interviewed. The second and third columns indicate whether the employers in the given group gave any Offers and Ranks, respectively (we define *Top Ranks* to be ranks of two or three). The next column shows the percentage of employers assigned to each group (e.g., the first row indicates that 5% of employers did not give any Ranks or Offers). The next column reports the percentage of employers that were *not* matched with any students by the algorithm, labelled "%NoMatch"; clearly, employers who did not give any ranks or offers have no-match rate of 100%. The next column, "%MatchR1", shows the percentage of employers that were matched with their first choice and the average evaluation score the employers gave to these students (higher is better). Finally, the last column, "%MatchR>1", shows the percentage of employers that were matched with a student who was not their first choice and the average evaluation score the employers gave to those students. We will discuss the percentages further in Section 3.2.

Note that the sum of the percentages reported in the last three columns – "%NoMatch" plus "%MatchR1" plus "%MatchR>1" – is 100 for each row. In other words, there are three possible options for employers: does not match with any student, matches with their first choice, or matches with their not-first choice.

To characterize employers with different ranking strategies, we inspected their names and job titles (Step 3 of Figure 2). We found that employers who gave:

- "No Offer/s or Rank/s" (first row of Table 2) consisted of companies of all sizes and industries, mainly offering "analyst" and "assistant" positions.
- "Only Rank/s" (second row) were mainly business units of the institution, and mostly offered "analyst", "support", and "intern" positions.
- "Only Offer/s" (third row) consisted of large well-known technology and manufacturing companies, offering "software developer" and "design" positions.
- "Offer/s and Top Rank/s" (fourth row) consisted of (a) medium-sized companies offering positions in "software development" and "data science", (b) large companies with positions such as "application development", "UI designer", "quality assurance", and "process improvement", and (c) companies with specialized jobs in the fields of electrical engineering, hardware, medical engineering, banking, etc.
- "Offer/s and Other Rank/s" (fifth row) consisted of small to medium-sized companies with job titles including "quality assurance", "software testing", "support technician", and "systems administrator".

3.2 Consequences of Employer Ranking Strategies

This section analyzes how ranking strategies used by employers affect their chances of finding a match and whether employers with different ranking strategies evaluate their matches differently at the end of the workterm. To provide context, we start by reporting the

Table 2: Employer ranking strategies

Label	Offer/s	Rank/s	%	%NoMatch	%MatchR1 (Avg(Eval))	%MatchR>1 (Avg(Eval))
No Offer/s or Rank/s	No	No	5	100	-	-
Only Rank/s	No	Yes	9	81	-	19 (5.9)
Only Offer/s	Yes	No	27	48	52 (6.1)	-
Offer/s & Top Rank/s	Yes	Yes (all r$\leq$3)	48	31	46 (6.1)	24 (5.9)
Offer/s & Other Rank/s	Yes	Yes (some r>3)	11	20	48 (6.2)	32 (5.9)

matching percentage and evaluation scores averaged across all the employers who participate in the ranking process (i.e., those who give at least one non-zero rank).

Overall, 39% of employers who participate in the ranking process do not find a matching student. Out of the 61% who find a match, 75% match with their first choice (i.e., with a student to whom they gave an Offer), and 25% match with their >1 choice. On average, employers who match with their first choice evaluate their students slightly higher (6.1) than those who match with their >1 choice (5.9). This difference is statistically significant at a p-value of 0.05.

Next, we analyze the consequences of employer ranking strategies on matching and evaluation. Recall that Table 2 shows the percentage of employers with different ranking strategies who do not find a match, match with their first choice (i.e., with a student to whom they gave an Offer), and match with their >1 choice. For each ranking strategy, we also show the average evaluation scores given by employers to their students. Among employers who make offers, those who provide more backup options (Offers and Ranks) have a higher matching rate, with a greater proportion matching with their backup choice. Additionally, one-fifth of the employers who "Only Rank" (i.e., do not make any offers and only use ranks of two and above) find a match.

On average, regardless of the ranking strategy used, employers who match with their first choice evaluate their students similarly and so do employers who match with their >1 choice. In addition, irrespective of the ranking strategy used, employers who match with their first choice evaluate their students slightly higher than those who matched with their >1 choice. Therefore, while employer ranking strategy affects their chances of finding a match, employers with different ranking strategies do not evaluate their students differently.

4. DISCUSSION AND CONCLUSIONS

In this paper, we proposed a new way of characterizing competition in a co-operative job market by studying how employers rank students after a round of interviews. Based on a dataset from a large co-operative education program, we identified ranking strategies, studied the characteristics of employers who use different strategies, and analyzed the effects of ranking strategies on matching and workterm evaluation. Our main findings are as follows.

Ranking strategies characterize competition: Ranking strategies may be used to characterize the extent of competition in the co-op job market; employers appear to be aware of the competition they face and rank accordingly.

Large employers are less likely to provide backup options. Thus, it appears that these employers are confident in their ability to hire their top choices, and if their top choices decline the offers, these employers are willing to risk not hiring any students from this university. Small to medium companies, especially those offering entry-level positions, are more likely to provide backup options, perhaps as a consequence of perceived competition for their top choices. Therefore, an employer's popularity and quality of job they offer appear to be correlated with the ranking strategy they use. Similar observations were made in many competitive environments, including supply chains and legal contracting, where parties with more bargaining power leverage their reputation when negotiating with others [9, 15].

Rank of match affects satisfaction: Regardless of the ranking strategy, employers who match with their first choice evaluate their co-op students slightly higher on average than those who match with their backup choices. In other words, satisfaction only seems to depend on the rank of the match and not on the strategy used to obtain the match.

These results should be interpreted carefully since they are based on data from a single institution. However, the methodology we presented in this paper may be used by others to reflect the extent of competition in their institutions through ranking patterns. In addition, this study is limited to identifying frequent patterns in the data, but not cause-and-effect relationships. This provides a starting point for further study: interviewing students and employers about the competition they face in the co-op market is an interesting direction for future work. Nevertheless, we believe that our findings will be of interest to students, employers and the institution. We provide several examples of actionable insights below.

1. Our results can help students understand how co-op employers rank their options in various situations. This may inform students' strategies and decision-making during applications and ranking, in turn, increasing their chances of finding a suitable co-op job.
2. Our results can inform new employers about the extent of competition in the co-op market, which in turn can help them decide how to rank their options given the competition they are likely to face.
3. Our findings indicate that some employers are confident in their ability to hire their top choices, indicating that such jobs are highly sought after by students. The institution may consider recruiting more such employers. On the other hand, the institution may recommend smaller employers to less-experienced students to increase their chances of finding a match.
4. Our findings suggest that employers who match with a backup choice are less satisfied with their co-op students. This suggests a need for methods to help manage the expectations of employers and students in this situation.

5. REFERENCES

[1] A. E. Barber and M. V. Roehling. Job postings and the decision to interview: A verbal protocol analysis. *Journal of applied psychology*, 78(5):845, 1993.

[2] J. M. Barron, J. Bishop, and W. C. Dunkelberg. Employer search: The interviewing and hiring of new employees. *The Review of Economics and Statistics*, pages 43–52, 1985.

[3] C.-F. Chien and L.-F. Chen. Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry. *Expert Systems with applications*, 34(1):280–290, 2008.

[4] S. Chopra, A. Khan, M. Mirsafian, and L. Golab. Gender differences in work-integrated learning assessments. In *Proceedings of the International Conference on Educational Data Mining (EDM)*, pages 524–527, 2019.

[5] S. Chopra, A. Khan, M. Mirsafian, and L. Golab. Gender differences in work-integrated learning experiences of stem students: From applications to evaluations. *International Journal of Work-Integrated Learning*, 21(3):253–274, 2020.

[6] A. Cuevas. Linkedin: Global recruiting trends 2016. `https://business.linkedin.com/content/dam/business/talent-solutions/global/en_us/c/pdfs/GRT16_GlobalRecruiting_100815.pdf`. Accessed: 19th Feb, 2021.

[7] Y. Jiang and L. Golab. On competition for undergraduate co-op placements: A graph mining approach. In *Proceedings of the International Conference on Educational Data Mining (EDM)*, pages 394–399, 2016.

[8] Y. H. Jiang, S. W. Y. Lee, and L. Golab. Analyzing student and employer satisfaction with cooperative education through multiple data sources. *Asia-Pacific Journal of Cooperative Education*, 16(4):225–240, 2015.

[9] D. Kennedy. Distributive and paternalist motives in contract and tort law, with special reference to compulsory terms and unequal bargaining power. *Md. L. Rev.*, 41:563, 1981.

[10] M. S. Parsa and L. Golab. Social media mining to understand the impact of co-operative education on mental health. In *Proceedings of the International Conference on Educational Data Mining (EDM)*, pages 653–657, 2020.

[11] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, and H. Xiong. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–34, 2018.

[12] S. L. Rynes, R. D. Bretz Jr, and B. Gerhart. The importance of recruitment in job choice: A different way of looking. *Personnel psychology*, 44(3):487–521, 1991.

[13] G. R. Thiel and N. T. Hartley. Cooperative education: A natural synergy between business and academia. *SAM Advanced Management Journal*, 62(3):19–25, 1997.

[14] A. Toulis and L. Golab. Graph mining to characterize competition for employment. In *Proceedings of the 2nd International Workshop on Network Data Analytics*, pages 1–7, 2017.

[15] J. Webster. Networks of collaboration or conflict? electronic data interchange and power in the supply chain. *The Journal of Strategic Information Systems*, 4(1):31–42, 1995.

[16] C. Zhu, H. Zhu, H. Xiong, C. Ma, F. Xie, P. Ding, and P. Li. Person-job fit: Adapting the right talent for the right job with joint representation learning. *ACM Transactions on Management Information Systems (TMIS)*, 9(3):1–17, 2018.

AQuAA: Analytics for Quality Assurance in Assessment

Manqian Liao, Yigal Attali, Alina A. von Davier
Duolingo, Inc.
mancy@duolingo.com, yigal@duolingo.com, avondavier@duolingo.com

ABSTRACT

High-stakes digital-first assessments are assessments that can be taken anytime and anywhere in the world and their scores impact test takers' lives. Computational psychometrics, a blend of theory-driven psychometrics and data-driven algorithms, provides the theoretical underpinnings for these data-rich assessments. The unprecedented flexibility, complexity, and high-stakes nature of these digital-first assessments poses enormous quality assurance challenges. In order to ensure these assessments meet both "the contest and the measurement" requirements of high-stakes tests [5], it is necessary to conduct continuous pattern monitoring and be able to promptly react when needed. In this paper, we illustrate the development of a quality assurance system, Analytics for Quality Assurance in Assessment (AQuAA), for a high-stakes and digital-first assessment. To build the system, educational data from continuous administrations of the assessments are mined, modeled and monitored via an interactive dashboard.

Keywords

high-stakes assessment, digital-first assessment, quality assurance

1. INTRODUCTION

Digital-first assessments are based on artificial intelligence (AI) tools that direct and optimize test-takers' experience. These digital tools include automatic systems for test development, scoring, and test delivery. In contrast to traditional large-scale assessments that are based on in-person administration to large groups of test takers in fixed locations, digital-first assessments are administered continuously to individual test takers, thus allowing for unprecedented flexibility. The advantages of the digital-first assessments have manifested themselves during the pandemic when traditional group assessments in brick-and-mortar test centers became impractical.

Manqian Liao, Yigal Attali and Alina A. von Davier "AQuAA: Analytics for Quality Assurance in Assessment". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 787-792. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

When digital-first assessments are used for high-stakes purposes (for example, for admissions or employment purposes), they, as any traditional high-stakes assessments, have a significant potential impact on test takers' lives. Thus, the digital-first high-stakes assessment also need to meet both "the contest and the measurement" requirements of high-stakes tests [5], where the "contest" here refers to the expectation that the test gives everyone a fair chance; the "measurement" refers to the requirement that the test is accurate and valid.

Quality assurance refers to a systematic process to maintain the high quality of the test and assessment scores and to prevent errors from all stages of the test, including test design, item design and development, test scoring, test analysis and score reporting [7]. Its complement, quality control, refers to a set of methods and statistics to evaluate the quality of the test. Many of the statistics and methods employed for quality assurance and quality control are similar, with quality control being part of the quality assurance overarching system. The International Test Commission Guidelines have articulated step-by-step procedures for quality control of general educational assessments but many of the steps are more applicable to traditional assessments, that is, "large-scale testing operations where multiple forms of tests are created for use on set dates."[7]

Since digital-first assessments differ from traditional assessments in many respects (e.g., administration frequency, item bank size), it is necessary to develop quality assurance procedures that are tailored for digital-first assessments. Developing such systems also requires research into the appropriate methodology to identify the most relevant statistics to be monitored for such new type of assessment, which is the focus of this paper.

In order to conduct quality assurance for digital-first high-stakes assessments, we developed a monitoring system named Analytics for Quality Assurance in Assessment (AQuAA), which is a blend of psychometrics and educational data mining packed into a dynamic and interactive dashboard-based system. AQuAA was designed to accommodate at least two unique characteristics of the digital-first assessments. On one hand, many key aspects of digital-first assessments, such as item generation and scoring, are automatically accomplished by machine. Therefore, compared to traditional assessments, the quality assurance of the digital-first assessments requires more extensive data mining techniques.

Computational psychometrics [18, 17] is leveraged to mine and model educational data in order to develop the statistics included in AQuAA. On the other hand, as a consequence of the continuous nature of administration, the quality assurance activities for digital-first assessments need to be conducted more frequently with a flexible timeline. In addition, tools that facilitate swift and efficient communication are indispensable so that prompt actions can be taken when issues are detected. In AQuAA, a variety of statistics are updated regularly and are integrated into an interactive dashboard for continuous pattern monitoring and timely communication purposes. AQuAA is also symbiotic with other activities (such as item development) given the fact that conclusions drawn from AQuAA could be used to direct the maintenance and improvement of the assessment.

This paper elaborates the development of AQuAA and aims to address three research questions: 1) What statistics should be used as indicators of test quality and score validity of digital-first assessments? 2) How to identify patterns and irregularities relevant to test quality of digital-first assessments? and 3) How to communicate the findings from the quality assurance process to stakeholders? This paper is focused on the the quality assurance of the test administration activities.

2. RELATED WORK

Quality assurance plays an important role in maintaining test score validity. [1] indicated that mistakes that jeopardize the assessment score validity could occur at all stages of assessment development and administration and that the mistakes could accumulate since many stages are contingent on previous stages. Therefore, quality control guidelines and step-by-step procedures [1, 2, 7] have been developed to help test developers identify possible mistakes as well as the causes of these mistakes, thereby helping them to identify solutions to fix the mistakes and prevent the mistakes from happening again.

Quality control procedures were mostly designed for traditional large-scale assessments that are administered in only a few test dates and have large test volumes in each administration [7, 1], with [2] being an exception. [2] recommended a quality control procedure for continuous mode tests (i.e., tests that are administered to small groups of test takers on many test dates) which share some similarities with digital-first assessments. Moreover, [2] have demonstrated an automated quality control system for continuous mode tests and the system consists of both an automatic part and a human review part. These two parts also apply to the quality assurance of digital-first assessments. In the automatic part, a number of steps that need to be conducted recurrently and can be implemented programmatically are packed into an automatic procedure with the use of digital tools. Steps in such an automatic procedure may include fetching the data from the database, conducting a variety of quality control analyses (see [9] for a review of quality control methods) and generating statistical reports. In the human review part, human experts are trained to review the statistical reports generated from the automatic procedure in order to identify potential irregularities or outliers, and determine whether or what actions need to be taken to handle these irregularities.

The foundation of an automated quality assurance procedure consists of a wide range of data mining and data visualization techniques. In the realm of quality assurance, the data mining and data visualization techniques serve two major purposes: First, to describe the trends and seasonal patterns of the assessment statistics; Second, to detect abrupt changes in the relevant assessment statistics. [9] have summarized a number of statistical methods and data visualization techniques for score quality assurance purposes. Various time series techniques can be chosen to describe trends or seasonal patterns, which include linear ANOVA models [4], regression with autoregressive moving-average [10], harmonic regressions [8] and dynamic linear models [19]. The Shewhart chart is a useful data visualization tool for continuous of the test score characteristics [9, 12, 14]. In terms of detecting abrupt changes in the assessment statistics, some model-based approaches have been applied to mine the data and identify abrupt changes in score time series, such as change-point models and hidden Markov model [9]. A data visualization techniques for detecting abrupt changes is cumulative sum (CUSUM) charts [13].

The products of the automated quality assurance procedure may include summary tables of the statistics, graphs and statistical testing results [2]. These statistical products could be organized into different formats, such as reports [2] and dashboards [11]. Since the products of the automated quality assurance procedure will serve as the starting point of the human review process [2], the choice of organizing format should be determined by the ease of communication to the targeted stakeholders.

3. MAJOR COMPONENTS OF AQUAA

This section illustrates how several key components of AQuAA address the research questions mentioned above. AQuAA has been launched as a minimum viable product (MVP) and additional features and statistics are being added to the system. This paper demonstrates the application of AQuAA the Duolingo English Test, a digital-first assessment. In order to help readers understand the context from which the AQuAA is developed, this section will start with a brief overview of the Duolingo English Test. However, the methodologies for designing AQuAA and the statistics considered for evaluation are intended to be adaptable to other digital-first assessments.

3.1 Overview of the Assessment

The Duolingo English Test is a high-stakes computerized adaptive test that is designed to be accessible anywhere and anytime [15]. Thus, it also falls under the category of continuous mode assessments [2]. The Duolingo English Test is an adaptive test, with a very large item bank that has been designed by subject matter experts (SMEs) and produced automatically by the machine. The items are reviewed by panels of SMEs to ensure quality and cultural fit. The items are scored automatically and the scoring methods are reviewed periodically by SMEs. Each individual test is proctored remotely using a complex and innovative asynchronous system that involves both AI-based tools and human proctors. Discrepancies or unusual situations are adjudicated by SMEs. Test results are reviewed through the quality assurance process in AQuAA. As part of this process, a wide range of process information related to test takers' behavior

Figure 1: AQuAA updating procedure

(e.g., time per item response, length of responses, etc.) is analyzed and monitored for quality assurance. The amount of data and the multiple sources and types of data are significantly more demanding of sophisticated analytics than is the case in more traditional assessments.

3.2 Overview of AQuAA

An overview of the procedure of developing and updating AQuAA is shown in Figure 1. Except for the first step (i.e., importing the data) that is relatively straightforward, the design of each step requires deliberation and, thus, is elaborated in the following sections. The steps in Figure 1 are scheduled to be automatically implemented on a daily basis (and in some cases more frequently). R [16] is the major programming tool used to develop AQuAA and automate the AQuAA updating process.

3.3 Checking and Cleaning Data

In general, the assessment data used for AQuAA can be separated into two types: Person-level data and item-response-level data. Person-level data contain variables that describe the overall person/session information, such as test takers' overall test score, sub-scores, test dates, and background characteristics. Item-response-level data contains variables that delineate information about each item the test taker responded to, such as item IDs, item difficulty levels, item responses and item scores, and other process information such as time duration test takers spent on each item.

After the data are imported, the integrity of the data is inspected to ensure that the data used for subsequent analyses are accurate and of high quality. For example, data are inspected for irregular values (e.g., negative values in time duration variables), and the causes of any such values are further investigated to identify any potential threats to the integrity of the data collection process.

3.4 Tracking Metrics and Statistics

The first research question is to determine what metrics and statistics are most relevant to monitor over time in order to evaluate the health of a continuous assessment. In order to support a statistical quality assurance system, AQuAA monitors results in the following five categories across time, adjusting for seasonality effects.

1. **Scores**. Test scores are directly used by test users (e.g., test takers, institutions), thus important indices at the level of test scores, including overall scores, sub-scores, and item type scores, are tracked in AQuAA. Score-related statistics include the location and spread of scores, inter-correlations between scores, bivariate or multivariate outliers, person fit, internal consistency reliability measures and standard error of measurement (SEM), and validity coefficients (e.g., correlation with self-reported external measures).

2. **Test taker profile**. The composition of the test taker population is tracked over time, as it could be used to explain the variability in test scores to some extent. Specifically, the (percentage) volume of test takers in the important population categories, such as country, native language, gender, age, intent in taking the test, and other background variables, are tracked. In addition, many of the score statistics are tracked across major test taker groups.

3. **Repeaters**. Repeaters are defined as those who take the test more than once within a 30-day[1] window. The prevalence, composition, and performance of the repeaters are tracked. The composition of the repeater population is defined with respect to the same test taker profile categories discussed above. The performance of the repeater population is tracked with many of the same test score statistics identified above, with additional statistics that are specific to repeaters: location and spread of both the first and second tests, as well as their difference, and test-retest reliability (and SEM).

4. **Item analysis**. As tests consist of items, ensuring that items are of high quality and that the item quality is stable over time are the prerequisites of maintaining the validity of the test scores. In AQuAA, item quality is quantified with four categories of item performance statistics: Item difficulty, item discrimination, item slowness (response time), and differential item functioning (DIF). Tracking these statistics would help test developers to develop expectations about the item bank with respect to item performance, flag items with extreme and/or inadequate performance, and detect drift in measures of performance across time.

5. **Item exposure**. The item exposure statistics concern how frequent each item (or each group of items) are used. An item being used either too frequently (over-exposure) or too infrequently (under-exposure) are undesirable for maintaining the item quality. An important statistic in this category is the item exposure rate, which is calculated as the the number of test administrations containing a certain item divided by the total number of test administrations. Tracking the item exposure rates can help flag under- or over-exposure of items.

3.5 Identifying Patterns and Irregularities

The second research question concerns the identification of patterns and irregularities in the data, which involves the development of the alarming mechanism of AQuAA. Developing the alarming mechanism in AQuAA is challenging partly due to the fact that the population of test takers is evolving and changing constantly, and, thus, many of the tracked metrics cannot be assumed to be stationary over time. Instead, the tracked metrics are often prone to systematic variation over and beyond predictable changes due to seasonality effects, thereby making it complicated to set an appropriate alarming criteria for the alarming mechanism.

[1]The day threshold of determining repeaters could be adjusted based on the test taking policy and the research purpose

The alarming mechanism in AQuAA is intended to detect persistent but smaller trends as well as alert large and abrupt changes that may be due to a problem in the assessment. To achieve these goals, we combined model-based psychometric analyses method with the time series and control charts techniques, both of which are useful for distinguishing systematic changes from chance variation in outcome processes.

The psychometric model-based methods allow us to track metrics after adjusting for certain factors (e.g., test takers' background characteristics), thus increasing the metrics' comparability over time. Specifically, in AQuAA, the item statistics and metrics are adjusted for test taker ability and background variables, and test taker statistics and metrics are adjusted for item characteristics.

3.6 Communicating Results

Our third research question involves how to communicate the information to the operational analysts as well as to the business unit. To visualize the trends and patterns of the statistics and facilitate the communication in the human review process, statistics are plotted using the ggplot2 R package [20]. Line plots are one of the most basic tools to visualize the time-series data. For example, Figure 2 demonstrates the stable trend of the mean of the overall test score during the Fall of 2020. Each dot in these figure represent a statistic calculated using a day worth of data; the lines are smoothed lines created by the locally weighted scatterplot smoothing (LOWESS) [3] method in order to represent the trends of the statistics.

Plots are also used to visualize the alerts raised by the AQuAA alarming mechanism introduced in Section 3.5. In AQuAA, the alerts are classified into three severity categories which are represented by different color codes. Specifically, yellow, orange and red represent low, medium and high levels of severity, respectively. For example, Figure 3 displays a monitoring plot for the daily median response time a few alerts in low severity. Once an alert is raised by AQuAA, messages are automatically sent to inform all the relevant stakeholders via email and the organization communication tool.

Various statistics and figures are integrated into an interactive dashboard using the flexdashboard [6] package. Figure A.1 demonstrates the layout of the dashboard. At the top of the dashboard (i.e., Section 1), there are five tabs corresponding to the five categories of statistics articulated in Section 3.4. Within each tab, the relevant statistics are arranged into storyboards: The statistics could be further classified into subcategories and allocated into different pages (i.e., Section 2); figures are displayed at the major section of the dashboard (i.e., Section 3); text description and some numerical results are displayed in the commentary section (i.e., Section 4).

4. THE APPLICATION OF AQUAA

As the quality assurance of digital-first assessments is a combination of automatic processes and human review processes, the AQuAA system is used as the starting point for the human review process, and the human review process, in turn, helps AQuAA to evolve into a more powerful tool to detect assessment validity issues. Figure B.1 demonstrates

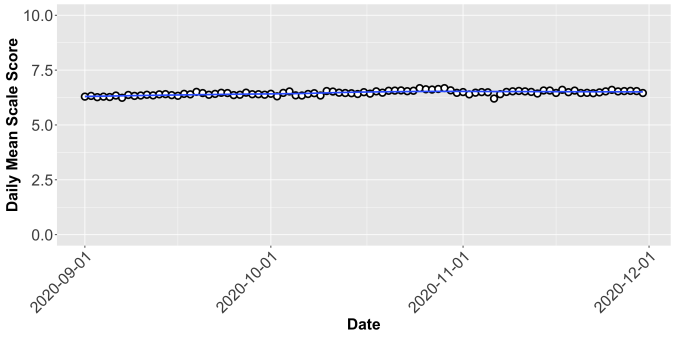


Figure 2: Trend of daily mean overall scores

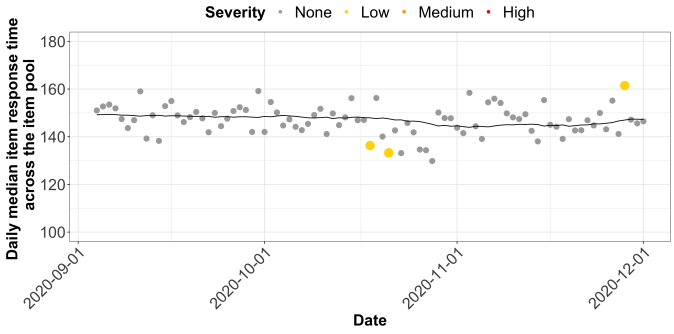


Figure 3: Trend of daily median response time with alerts.

an example human review process following every week's updates of AQuAA: SMEs meet to review the alerts raised by AQuAA alarming mechanism and review for any anomalies that are suggested by the AQuAA figures but have not been caught by the AQuAA alarming mechanism. The SMEs review each individual alert and determine whether it is an actual sign of a validity issue or it is a false alarm. If the alarm is believed to be caused by a validity issue, follow-up actions are taken to determine the severity and urgency, fix and document the issue. If the issue had not been caught by the AQuAA alarming mechanism, improvements would be made to the AQuAA functionality such that AQuAA would be more sensitive in detecting the issue.

5. DISCUSSION

This paper demonstrates the development of a quality assurance system that is tailored for digital-first assessments that are continuously administered. Several research questions motivated many of these approaches, as very few of the traditional methods apply to the digital-first assessments. The steps and considerations for building the quality assurance system have been elaborated, so that test developers could adapt the methodologies in this paper to their own assessments. It should be noted that the list of quality assurance statistics presented here is not exhaustive. Instead, due to the data-rich nature of the digital-first assessment, the list of monitoring statistics is expected to be lengthened and improved as the research in statistical techniques advances. The list of monitoring statistics should also be customized to the purposes and characteristics of the assessment. Hence, the infrastructure of AQuAA is designed to be so flexible as to incorporate and monitor additional statistics.

6. REFERENCES

[1] A. Allalouf. Quality control procedures in the scoring, equating, and reporting of test scores. *Educational Measurement: Issues and Practice*, 26(1):36–46, 2007. Publisher: Wiley Online Library.

[2] A. Allalouf, T. Gutentag, and M. Baumer. Quality Control for Scoring Tests Administered in Continuous Mode: An NCME Instructional Module. *Educational Measurement: Issues and Practice*, 36(1):58–68, 2017. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/emip.12140.

[3] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979. Publisher: Taylor & Francis.

[4] S. J. Haberman, H. Guo, J. Liu, and N. J. Dorans. Consistency of SAT® I: Reasoning test score conversions. *ETS Research Report Series*, 2008(2):i–20, 2008. Publisher: Wiley Online Library.

[5] P. W. Holland. Measurements or contests? Comments on Zwick, bond and Allen/Donoghue. In *Proceedings of the social statistics section of the American Statistical Association*, volume 1994, pages 27–29. American Statistical Association Alexandria, VA, 1994.

[6] R. Iannone, J. J. Allaire, B. Borges, RStudio, K. I. D. CSS), A. A. D. CSS), J. Mosbech (StickyTableHeaders), N. Bossart (Featherlight), L. Verou (Prism), D. Baranovskiy (Raphael.js), S. Labs (Raphael.js), B. Djuricic (JustGage), T. Sardyha (Sly), B. Lewis (Examples), C. Sievert (Examples), J. Kunst (Examples), R. Hafen (Examples), B. Rudis (Examples), and J. Cheng (Examples). flexdashboard: R Markdown Format for Flexible Dashboards, June 2020.

[7] International Test Commission (ITC). ITC Guidelines on Quality Control in Scoring, Test Analysis, and Reporting of Test Scores. *International Journal of Testing*, 14(3):195–217, July 2014. Publisher: Taylor & Francis Ltd.

[8] Y.-H. Lee and S. J. Haberman. Harmonic regression and scale stability. *Psychometrika*, 78(4):815–829, 2013. Publisher: Springer.

[9] Y.-H. Lee and A. A. von Davier. Monitoring scale scores over time via quality control charts, model-based approaches, and time series techniques. *Psychometrika*, 78(3):557–75, 2013.

[10] D. Li, S. Li, and A. A. von Davier. Applying time-series analysis to detect scale drift. In *Statistical models for test equating, scaling, and linking*, pages 327–346. Springer, 2009.

[11] L. Mohadjer and B. Edwards. Paradata and dashboards in PIAAC. *Quality assurance in education*, 2018. Publisher: Emerald Publishing Limited.

[12] M. H. Omar. Statistical process control charts for measuring and monitoring temporal consistency of ratings. *Journal of Educational Measurement*, 47(1):18–35, 2010. Publisher: Wiley Online Library.

[13] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954. Publisher: JSTOR.

[14] W. D. Schafer, B. J. Coverdale, H. Luxenberg, and J. Ying. Quality control charts in large-scale assessment programs. *Practical Assessment, Research, and Evaluation*, 16(1):15, 2011.

[15] B. Settles, G. T. LaFlair, and M. Hagiwara. Machine Learning–Driven Language Assessment. *Transactions of the Association for Computational Linguistics*, 8:247–263, 2020.

[16] R. D. C. Team. R: A language and environment for statistical computing. 2013.

[17] A. A. von Davier. Virtual and collaborative assessments: Examples, implications, and challenges for educational measurement. In *Invited Talk at the Workshop on Machine Learning for Education, International Conference of Machine Learning 2015*, 2015.

[18] A. A. von Davier. Computational psychometrics in support of collaborative educational assessments. *Journal of Educational Measurement*, 54(1):3–11, 2017.

[19] R. G. Wanjohi, P. W. van Rijn, and A. A. von Davier. A state space approach to modeling irt and population parameters from a long series of test administrations. In *New developments in quantitative psychology*, pages 115–132. Springer, 2013.

[20] H. Wickham. ggplot2. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(2):180–185, 2011. Publisher: Wiley Online Library.

APPENDIX

A. DEMO OF AQUAA

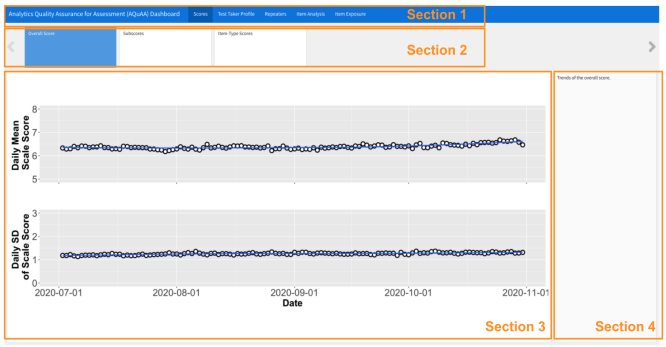


Figure A.1: Demo of AQuAA with annotations. Section 1 is the navigation bar containing five tabs corresponding to the five categories of statistics monitored in AQuAA. Within each tab, the relevant statistics are grouped into subcategories and are arranged into storyboards. Section 2 display the pages that correspond to the subcategories of statistics. Section 3 is the major section of the dashboard where figures are displayed. Section 4 is the commentary section that display the text description and numerical results.

B. SUBJECT MATTER EXPERT REVIEW PROCESS

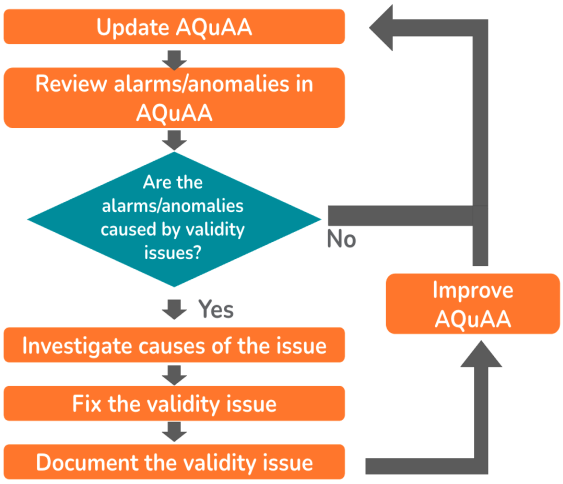


Figure B.1: Subject Matter Expert (SME) review process.

LMS Log Data Analysis from Fully-Online Flipped Classrooms: An Exploratory Case Study via Regularization

Jin Eun Yoo
Korea National University of Education
jeyoo@knue.ac.kr

Minjeong Rho
Korea National University of Education
minjeong019@gmail.com

ABSTRACT

This study illustrated an exploratory study of LMS log data from undergraduate fully-online flipped classrooms. A total of 237 students' instructional video watching behaviors were extracted from LMS, and were analyzed with background variables to predict students' final performance. Regularization was proposed a suitable machine learning technique, as it produces interpretable prediction models. Specifically, Enet (elastic net) and Mnet were employed to handle possible multicollinearity in LMS log data, and the prediction models of Enet and Mnet identified 19 and 21 important predictors of final performance out of 157, respectively. In particular, both regularization models were able to screen lower-performing students as early as the first week of the course. Mere attempts to watch difficult videos after class increased the final scores.

Keywords

LMS log data, machine learning, regularization, flipped classroom, performance modeling

1. INTRODUCTION

The COVID-19 pandemic has changed the education system worldwide. Online learning is no longer an option, and an increasing number of online classes incorporate components of flipped classrooms (FC) in an effort to improve the quality of learning and instruction. Despite varying results regarding the effectiveness of flipped learning in higher education [1, 2, 3, 4], FC has grown rapidly as an innovative pedagogical approach in recent decades. In FCs, students' active involvement in pre-class activities is greatly emphasized as a necessary condition to enhance in-class learning and instruction [5]. However, there has been little empirical research on whether students completed the assigned pre-class activities and whether pre-class activities lead to desired outcomes.

This may relate to analytical limitations of the previous research in terms of data and methods. First of all, learning management system (LMS) log data are a crucial source of information in order to capture students' learning activities. However, not all the studies on FC collected data from LMS, particularly when pre-class assignments do not involve activities in LMS. For instance, the assignment of reading materials cannot be properly recorded outside of LMS. Researchers can ask students ex post facto in a self-report survey. However, self-report questionnaires rely on memories and reflections, and thus are prone to social desirability bias. On the other hand, LMS log data unobtrusively collect near-real-time information; students' activities in LMS are automatically stored in the log files without the students' cognizance [6, 7, 8, 9]. Particularly in the COVID-19 situation, fully-online FCs has emerged. In fully-online FCs both pre-class and in-class activities take place online using platforms such as LMS, and therefore collecting trace data has become much easier than in the original FCs.

Next, there is room for improvement in terms of analysis methods. Despite the aforementioned advantages that log data bring to data analyses, the intractability of log data has been a practical hindrance. Log data are unstructured, which can lead to high-dimensional data (i.e., more variables than observations), depending on data pre-processing and cleaning. Previous research on LMS log data to model students' achievement have analyzed students' behavioral data (e.g., instructional video watching behaviors) [6, 10, 11, 7, 12] as well as background (e.g., gender) [10, 13] and exam data [11, 13]. In particular, behavioral data were used as a tool to measure students' self-regulated learning [6, 10, 11, 7, 13, 12, 14, 15, 16], but aggregate variables such as total login frequencies or average login hours were analyzed with traditional methods [13, 15] or early ML (machine learning) techniques [14, 16]. As traditional methods are likely to result in nonconvergence problems with high-dimensional data, previous research may have used aggregate variables.

However, study time relevant to a specific instructional unit can be traced from log data, which will serve as a better indicator than the sum of study time, a crude measure of time investment in studying [15]. Such detailed information in turn will be conducive to understanding learning and instruction and giving specific, targeted, and timely feedback to students. This relates back to the issue of the previous LMS log data research: lack of empirical research on the relationship between pre-class assignments and students' performance at an instructional unit level. Particularly when behavioral variables at an instructional level are to be ana-

Jin Eun Yoo and Minjeong Rho "LMS Log Data Analysis from Fully-Online Flipped Classrooms: An Exploratory Case Study via Regularization". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 793-798. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

lyzed, ML is a necessary technique to analyze LMS log data from fully-online FCs.

Since completing pre-class assignments and preparing for interactive in-class activities is critical in FC, a high level of self-regulated learning (SRL) is necessary for students to succeed. SRL strategies related to students' academic success such as effective time management, metacognition, effort regulation, and critical thinking have been shown to have a significantly positive effect on students' academic success [17]. The question is which behaviors indicate SRL. Students carrying out SRL would naturally include more time on attending lectures and self-study which have a positive effect on academic achievement [18]. Previous studies have used variables such as login frequencies, LMS menu usage, material download, content pages viewed, and posted messages [6, 10, 11, 13, 14, 15, 16]. However, aggregate measures of these data display inconsistent effects on student achievement. For instance, login frequencies [13, 16] and LMS menu usage [13, 14, 15] were statistically significant or important indicators to students' academic achievement in online learning. In contrast, in MOOC (massive open online course) environments, forum variables such as numbers of messages posted, or comments received were found to be not directly related to students' learning [11].

Constant effort put into preparing for FCs may be difficult to capture with aggregate data. That is, instructional unit based log data would be a better predictor for academic success. A study predicting online student performance [19] demonstrates that the study habits of students with high levels of academic success can even be observed even in the first few weeks of a course. The implication is that instructional unit based analysis could yield richer information about the study patterns of students which eventually leads to timely intervention by the instructor.

Among ML, this study proposes regularization. Although 'prediction' is the operative word in ML, learning analytics is one of the fields which needs to be augmented with explanation. Regularization or penalized regression is known to produce explainable prediction models. Based on linear regression, the regression coefficients of regularization can be interpreted in the similar way as those in traditional, non-penalized regression. This is a great advantage in LMS data analysis, as prediction models need to be interpreted under certain educational settings, for instance to plan more effective intervention strategies for at-risk students. There has been little study employing regularization methods in LMS log data analysis. Specifically, this study chose Enet [20, 21] and Mnet [22] among regularization as they handle multicollinearity, a likely challenge in LMS data analysis. The two main research questions were as follows:

1. What are the students' instructional video watching behaviors like at an instructional unit level? Do students complete pre-class assignments in fully-online undergraduate flipped classrooms?

2. Among students' behavioral and background variables, which variables are important to predict students' academic achievement?

2. MACHINE LEARNING

For a Gaussian family, Enet and Mnet are expressed as equations 1 and 2, respectively. The second term on the right-hand side of equation 1 is the penalty function of Enet, consisting of two tuning parameters: λ and α. Enet is a combination of LASSO and ridge. The parameter λ regularizes shrinkage of the coefficients, and the parameter α controls the amount of ridge. When α is 1, equation 1 reverts to the LASSO equation, and when α is 0, it reverts to the ridge equation. Aforementioned, by adding the ridge component to the equation, Enet can handle multicollinearity.

$$\hat{\beta}^{\text{Enet}} = \operatorname{argmin}_{\beta} \left[\frac{1}{2n} \left\| y - \sum_{k=1}^{K} X_k \beta_k \right\|^2 + \lambda \sum_{k=1}^{K} \left(\alpha \left\| \beta_k \right\| + (1-\alpha) \left\| \beta_k \right\|^2 \right) \right]. \tag{1}$$

$$\hat{\beta}^{\text{Mnet}} = \operatorname{argmin}_{\beta} \left[\frac{1}{2n} \left\| y - \sum_{k=1}^{K} X_k \beta_k \right\|^2 + \sum_{k=1}^{K} J\left(\left\| \beta_k \right\| | \lambda_1, \gamma \right) + \lambda_2 \sum_{k=1}^{K} \left\| \beta_k \right\|^2 \right],$$
$$\text{where } J(\mathrm{x}|\lambda_1, \gamma) = \left\{ \begin{array}{ll} -\frac{1}{2\gamma} x^2 + \lambda_1 |x|, & |x| \le \gamma\lambda_1 \\ \frac{1}{2}\gamma\lambda_1^2, & |x| > \gamma\lambda_1 \end{array} \right\}. \tag{2}$$

Enet uses convex penalties, which increase linearly regardless of the coefficient size. By contrast, Mnet uses a concave penalty, which tapers off for coefficients in larger absolute values, yielding nearly consistent coefficient estimates[22]. Mnet has three tuning parameters (equation 2). The parameter λ_1 has the same regularization function as the λ penalty in Enet (equation 1). The γ parameter of Mnet controls the concavity of the convex penalty. When the concavity penalty goes to infinity, the MCP penalty reverts back to the LASSO penalty. Mnet also deals with multicollinear data; the penalty associated with λ_2 adds the ridge component to the equation.

To consider the bias resulting from data-splitting in model validation, this study executed subsampling techniques for variable selection [23, 24]. The following three steps were repeated 100 times with random data-splitting. First, the whole data were randomly divided with the ratio of 7:3 to get the training and test data, respectively. Second, for a value of the penalty parameter, the training data were split with the ratio of 4:1 to execute 5-fold CV. For a value of λ, the prediction error is calculated, which was referred to as the CV error of the λ [20]. Third, the second step was repeated for every λ in range, and the λ of the lowest CV error served as the penalty value of the regularization. That λ value was appplied to the test data in step 1, which yielded prediction measure.

The selection or non-selection of each variable from step 2 was counted in the 100 iterations, which served as the selection counts of the study. Particularly, this study presented

variables selected 1, 25, 50, 75 times or more, and all 100 times [25, 26]. All the programs were written in R 3.6.2. Specifically, the grpreg library [27] was used for regularization.

3. MATERIALS

In the Fall semester of 2020, 242 undergraduate students in pre-service teacher program enrolled in 8 fully-online undergraduate classes titled *Measurement and Evaluation*. The classes of the Fall semester were mandatory for sophomores majoring in Liberal Arts and Social Sciences. Three instructors (A, B, C) including a head-instructor (A) taught the 8 classes. All the 8 classes scheduled a simultaneous final at the end of the course, and shared the same class materials including instructional videos, textbooks, and syllabus. The instructional videos were pre-recorded PowerPoint presentations with the head-instructor talking, with content based on a book also written by the head-instructor. There were a total of 34 video clips covering 11 instructional topics in the corresponding 11 instructional weeks (refer to the videos 01_1 to 11_4 in Appendix A).

On the orientation day of the first week, the importance of the weekly assignments of instructional video watching before class were emphasized, particularly because students were asked to create and complete class projects within groups based on the contents of the assigned videos. During class, interactions in small groups of 4 to 6 students were greatly encouraged. The groups were engaged in discussions on team projects and SPSS exercises in Zoom breakout rooms. A non-mandatory quiz of 4-5 short questions was presented for each week in LMS. Students were told that the quizzes would serve as formative assessments and the quiz scores did not count toward grades.

In total, 21,589 rows of video watching activities as well as 5,107 rows on board-posts readings were recorded in the log file. As many of the students indicated that they used the double-speed option of the LMS in video watching, this study used 50 % of the video length as a criterion. If a student watched a video 50 % of the length or more, the student is counted to have completed watching the video, and vice versa.

As the first research question was to investigate students' video watching behaviors at an instructional unit level, this study counted the frequencies of each video, separating before/ after and attempted but incomplete/ completed video watching. Specifically, 4 variables were created for each video: BI (incomplete attempt before class), BC (complete watching before class), AI (incomplete attempt after class), and AC (complete watching after class). Six Aggregate variables were also obtained for comparison purposes to previous research: BI, BC, and B (before-combined (I+C)) for before class counts; and AI, AC, and A (after-combined (I+C)) for after class counts.

The response variable of this study was final. The final test consisted of 35 multiple-choice items, and was given simultaneously to all the 242 students at the last week of the course. There were 5 students who missed the final, and those students' data were excluded from further analysis. The background and response variable were merged to the variables from LMS data, which resulted in the final dataset of 157 predictors of 237 students.

4. RESULTS

4.1 Students' Video Watching Behaviors

Table 1 summarizes the descriptive statistics of students' instructional video watching behaviors. The 6 groups of cells present the summary results of the aggregate variables. Students watched the videos more often after class than before class. Throughout the course students on average attempted to watch and completed watching each video about 1.03 and 1.08 times after class, respectively, while the values dropped to 0.20 and 0.23 before class (Table 1). The mean values smaller than 1 indicate that the students on average did not watch all the videos. Attempts and completions combined (I+C), students on average clicked about half of the videos before class (0.42), but they clicked each of the videos more than twice after class (2.11).

The range of students' video watching frequencies was quite wide. Some students clicked none of the videos after class (AI min= 0.00), while others after class clicked and finished watching each video as many as 4.20 (AI max= 4.20) and 2.47 times (AC max= 2.47), respectively. The maximum frequencies of before class watching were also less than those of after class, 1.38 and 1.00 for incomplete and complete watching, respectively.

4.2 Machine Learning Results

4.2.1 RMSE and Selection Counts

RMSE (root mean square error) was the prediction measure of the response variable of this study. The RMSE averages of Enet and Mnet were 5.58 and 5.69 with SDs of 0.50 and 0.46, respectively.

Consistent with literature [28, 22], Mnet always selected fewer variables than Enet. Of note, 103 and 94 predictors were selected out of 157 at least once with Enet and Mnet, respectively. This signifies the importance of running multiple iterations and employing selection counts, particularly when the research purpose is variable selection via regularization [25, 26]. In other words, employing selection counts considers the bias resulting from random data-splitting in model building.

Applying 25 or more selection counts resulted in 33 and 21 predictors for Enet and Mnet, respectively. A total of 19 and 3 predictors were selected at least 1 out of 2 runs of Enet and Mnet, respectively. Four predictors were selected with 3 out of 4 runs of Enet, but there was no such predictor with Mnet and no predictor was selected in all the 100 iterations.

4.2.2 Selected Variables

This study on log data analysis presents the summary of predictors selected 50 or more for Enet and 25 or more for Mnet in Table 2. Due to space limit, part of the results are discussed. Student gender and grade were selected important. When the other variables were held constant, male students had lower final score than female students. Interestingly, on-grade students, sophomores, tended to have lower scores. Students' attitudes toward measurement and evaluation (attitudes) also resulted in higher scores.

Table 1: Students' Before and After Watching Frequencies per Video

	I (incomplete)				C (complete)				I+C (combined)			
	M	SD	min	max	M	SD	min	max	M	SD	min	max
B (before class)	0.20	0.18	0.00	1.38	0.23	0.08	0.06	1.00	0.42	0.21	0.08	1.62
A (after class)	1.03	0.78	0.00	4.20	1.08	0.38	0.00	2.47	2.11	1.00	0.06	6.24

Among variables extracted from log data, the total number of clicks on SPSS material postings (spss.sum) and the numbers of quiz-taking (test.M and test.P) were important predictors to final. More clicks on SPSS postings lead to higher scores on final. Specifically, one more click on the SPSS material increased students' final scores by 0.11 and 0.16 for Enet and Mnet, respectively. Similarly, although students knew that the scores on quizzes did not count toward the final grade, simply taking the quizzes increased the final scores regardless of the device (mobile or PC). Students who watched the instructional videos mobile (lecture.M) also tended to have higher scores in final.

Among the 142 variables on video watching, 12 to 13 variables were selected as important depending on the regularization method (Table 2). Findings from the 13 selected variables are as follows. First, the very first video turned out to convey crucial information in predicting students' achievement, although it covered the easiest contents on formative assessment. The more the students completed watching the first video before class (BC01_1), the higher their final scores were. Specifically, one more completed watching of the first video before class increased students' final score by 0.57 in Enet and by 0.68 in Mnet. By contrast, the more the students completed watching the first video after class (AC01_1), the lower the final scores were. One more completed watching of the first video after class decreased final scores by 0.45 and 0.53 in Enet and Mnet, respectively. This (AC01_1) was the only AC variable of negative relation to the final.

Second, with the exception of AC01_1, the other AC variables (e.g., AC03_1, AC03_2, AC04_3, AC09_3, AC10_2, AC11_4) had positive relations with the final. The selected AC variables covered either the earlier technical contents or the most difficult concepts at the end. Particularly, the earlier technical contents included the first SPSS practice (AC03_1, AC03_2) and Ebel and Angoff standard setting (AC04_3). Cronbach's alpha (AC09_3), reliability with SPSS (AC10_2), and the relationship between reliability and validity (AC11_4) covered the most difficult concepts in the last weeks of the course. Students who completed watching these videos multiple times after class were more likely to obtain higher final scores.

Third, the relationship of AI variables to the final seems to depend on the class progress. Students who attempted but failed to complete watching the video on Ebel and Angoff standard setting covered in the fourth instructional week had lower scores on final (AI04_3). By contrast, incomplete watching of some videos on the last topic (covered in the last instructional week) were positively related to the final (AI11_2 and AI11_4). Of note, both AC and AI variables on video 11_4, the last video, had positive correlation coefficients.

Table 2: Coefficients of Selected Predictors by Regularization

	variable	Enet			Mnet		
		mean	SD	#	mean	SD	#
1	gender	-0.54	0.35	65	-0.8	0.48	36
2	on-grade	-0.61	0.22	75	-0.75	0.31	56
3	test.M	0.34	0.12	62	0.48	0.15	38
4	test.P	0.23	0.10	75	0.31	0.15	49
5	lecture.M	0.01	0.01	51	0.02	0.02	30
6	attitudes	1.31	0.49	80	1.65	0.58	61
7	spss.sum	0.11	0.05	57	0.16	0.08	37
8	BC01_1	0.57	0.26	77	0.68	0.35	50
9	AC01_1	-0.45	0.17	68	-0.53	0.30	38
10	AC03_1	0.26	0.16	52	0.39	0.28	25
11	AC03_2	0.34	0.22	59	0.46	0.29	31
12	AI04_2	-0.19	0.08	67	-0.23	0.13	40
13	AC04_3	0.40	0.26	52	0.55	0.34	28
14	BI06_2	-0.46	0.18	54	-0.59	0.23	34
15	AC09_3	0.29	0.19	68	0.39	0.29	44
16	AC10_2	0.44	0.24	66	0.61	0.32	44
17	AI11_2	0.38	0.18	52	0.61	0.29	27
18	AC11_4	0.29	0.21	58	0.42	0.29	35
19	AI11_4	0.33	0.16	65	0.40	0.21	37
20	on-semester				1.39	1.09	25
21	AI04_3				-0.21	0.13	27

Note. # indicates the number of selection in 100 iterations.

5. DISCUSSION

This study predicted students' final scores with as few as 19 to 21 predictors out of 157 with regularization techniques. Of note, the prediction models of this study are explainable, as we employed regularization, which is based on linear regression. Specifically, Enet and Mnet were employed to handle multicollinear data. Surprisingly, the prediction models differentiated lower-performing students as early as the first instructional week, right after the orientation week. Instructors now can invest their efforts in intervention without waiting until a quiz or an exam. Completing difficult videos multiple times after class also lead to higher scores in the final. Moreover, mere attempts to watch them after class also increased the scores.

Despite its importance in FC, it has been a foggy area whether students completed the pre-class activities or not and whether the pre-class activities lead to desired outcomes. This study also contributed to partly uncover what was going on behind the curtain of FC. The students on average completed at most 1/5 of the videos before class. Stronger links need to be established between pre-class assignments and in-class team projects.

6. REFERENCES

[1] F. Chen, A. M. Lui, and S. M. Martinelli, "A systematic review of the effectiveness of flipped classrooms in medical education," *Medical Education*, vol. 51, no. 6, pp. 585–597, 2017.

[2] J. McGivney-Burelle and F. Xue, "Flipping calculus," *Primus*, vol. 23, no. 5, pp. 477–486, 2017.

[3] Y. Shi, Y. Ma, J. MacLeod, and H. H. Yang, "College students' cognitive learning outcomes in flipped classroom instruction: a meta-analysis of the empirical literature," *Journal of Computers in Education*, vol. 7, pp. 79–103, 2020.

[4] K. F. Hew and C. K. Lo, "Flipped classroom improves student learning in health professions education: A meta-analysis," *BMC Medical Education*, vol. 18, no. 38, 2018.

[5] J. Bergmann and A. Sams, *Flip Your Classroom: Reach Every Student in Every Class Every Day*. Washington, DC: International Society for Technology in Education, 2012.

[6] E. Popescu and F. Leon, "Predicting academic performance based on learner traces in a social learning environment," *IEEE Access*, vol. 6, pp. 72774–72785, 2018.

[7] E. Fincham, D. Gašević, J. Jovanović, and A. Pardo, "From study tactics to learning strategies: An analytical method for extracting interpretable representations," *IEEE Transactions on Learning Technologies*, vol. 12, no. 1, pp. 59–72, 2019.

[8] M. Manso-Vázquez, M. Caeiro-Rodríguez, and M. Llamas-Nistal, "An xAPI application profile to monitor self-regulated learning strategies," *IEEE Access*, vol. 6, pp. 42467–42481, 2018.

[9] A. A. ElSayed, M. Caeiro-Rodriguez, F. A. Mikic-Fonte, and M. Llamas-Nistal, "Research in learning analytics and educational data mining to measure self-regulated learning: A systematic review," in *The 18th World Conference on Mobile and Contextual Learning*, pp. 46–53, September 2019.

[10] R. Al-Shabandar, A. J. Hussain, P. Liatsis, and R. Keight, "Analyzing learners behavior in MOOCs: An examination of performance and motivation using a data-driven approach," *IEEE Access*, vol. 6, pp. 73669–73685, 2018.

[11] P. M. Moreno-Marcos, T. Pong, P. J. Muñoz-Merino, and C. Delgado Kloos, "Analysis of the factors influencing learners' performance prediction with learning analytics," *IEEE Access*, vol. 8, pp. 5264–5282, 2020.

[12] A. Montgomery, A. Mousavi, M. Carbonaro, D. V. Hayward, and W. Dunn, "Using learning analytics to explore self-regulated learning in flipped blended learning music teacher education," *British Journal of Educational Technology*, vol. 50, no. 1, pp. 114–127, 2019.

[13] J. W. You, "Identifying significant indicators using LMS data to predict course achievement in online learning," *Internet and Higher Education*, vol. 29, no. 1, pp. 23–30, 2016.

[14] M. Cho and J. S. Yoo, "exploring online students' self-regulated learning with self-reported surveys and log files: A data mining approach," *Interactive Learning Environment*, vol. 25, no. 5, pp. 51–65, 2017.

[15] L. Macfadyen and S. P. Dawson, "Numbers are not enough: Why e-learning analytics failed to inform an institutional strategic plan," *Educational Technology & Society*, vol. 15, no. 3, pp. 149–163, 2012.

[16] V. C. Smith, A. Lange, and D. R. Huston, "Predictive modeling to forecast student outcomes and drive effective interventions in online community college courses," *Journal of Asynchronous Learning Networks*, vol. 16, no. 3, pp. 51–61, 2012.

[17] J. Broadbent and W. L. Poon, "Self-regulated learning strategies & academic achievement in online higher education learning environments: A systematic review," *The Internet and Higher Education*, vol. 27, pp. 1–13, 2015.

[18] B. S. Grave, "The effect of student time allocation on academic achievement," *Education Economics*, vol. 19, no. 3, pp. 291–310, 2011.

[19] C. F. L. T. B. S. H. A. Sheshadri, N. Gitinabard, "Predicting student performance based on online study habits: A study of blended courses.," *arXiv preprint*, 2019.

[20] T. Hastie, R. Tibshirani, and J. Freedman, *The elements of statistical earning: Data mining, inference, and prediction*. New York: Springer, second ed., 2009.

[21] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of Royal Statistical Society Series B*, vol. 67, no. 2, pp. 301–320, 2005.

[22] J. Huang, P. Breheny, S. Lee, S. Ma, and C. Zhang, "The mnet method for variable selection," *Statistica Sinica*, vol. 26, no. 3, pp. 903–923, 2016.

[23] S. K. Shevade and S. S. Keerthi, "A simple and efficient algorithm for gene selection using sparse logistic regression," *Bioinformatics*, vol. 19, no. 17, pp. 2246–2253, 2003.

[24] N. Meinshausen and P. Bühlmann, "Stability selection," *Journal of the Royal Statistical Society: Series B*, vol. 72, no. 4, pp. 417–473, 2010.

[25] J. E. Yoo and M. Rho, "Exploration of predictors for korean teacher job satisfaction via a machine learning technique, group mnet," *Frontiers in Psychology*, vol. 11, no. 441, 2020.

[26] J. E. Yoo and M. Rho, "Large-scale survey data analysis with penalized regression: A Monte carlo simulation on missing categorical predictors," *Multivariate Behavioral Research*, 2021.

[27] P. Breheny and Y. Zeng, *Package 'grpreg'*, February 2019. `https://cran.r-project.org/web/packages/grpreg/grpreg.pdf`.

[28] J. Huang, S. Ma, and C. H. Zhang, "Adaptive lasso for sparse high-dimensional regression models," *Statistica Sinica*, vol. 18, no. 1, pp. 1603–1618, 2008.

APPENDIX

A. VIDEO IDS AND LABELS

	video ID	label
1	1_1	formative assessment
2	2_1	variables and scales
3	2_2	sampling
4	3_1	descriptive statistics
5	3_2	descriptive statistics (SPSS)
6	4_1	norm-referenced evaluation
7	4_2	criterion-referenced evaluation
8	4_3	Ebel and Angoff standard setting
9	5_1	measuring affective domains
10	5_2	observation
11	5_3	interviews
12	5_4	survey
13	6_1	performance assessment: definition
14	6_2	performance assessment: scoring
15	7_1	test construction steps
16	7_2	multiple-choice items
17	7_3	constructed-response items
18	7_4	scoring caveats
19	8_1	item difficulty and discrimination I
20	8_2	covariance and correlation
21	8_3	item difficulty and discrimination II
22	8_4	item difficulty and discrimination (SPSS)
23	9_1	introduction to reliability
24	9_2	types of reliability
25	9_3	Cronbach's alpha
26	9_4	standard error of measurement
27	9_5	factors influencing reliability
28	10_1	objectivity and reliability
29	10_2	reliability (SPSS)
30	10_3	objectivity (SPSS)
31	11_1	content validity
32	11_2	criterion-related validity
33	11_3	construct validity
34	11_4	reliability and validity

Measuring the Academic Impact of Course Sequencing using Student Grade Data

Tess Gutenbrunner, Daniel D. Leeds, Spencer Ross, Michael Riad-Zaky, and Gary M. Weiss
Computer and Information Science Department
Fordham University, New York, NY
{tgutenbrunner, dleeds, sross34, mriadzaky, gaweiss}@fordham.edu}

ABSTRACT

Undergraduate college students have substantial flexibility in choosing the order in which they take courses, since most courses either have no prerequisites or only a single prerequisite. However, the specific order that courses are taken can have an impact on student performance. This paper describes a general methodology for assessing the impact of course sequencing on student performance, as measured by course grades, and applies this methodology to eight years of undergraduate academic data from Fordham University. The results demonstrate that certain course orderings are associated with improved student grade performance. This study introduces a methodology, new metrics, and a publicly available data-processing tool that can be applied to any student course-grade data set to measure course sequencing effects. The results can be used to inform student decisions, modify course recommendations, and even modify course prerequisites.

Keywords

Data mining, education, course sequencing, student performance

1. INTRODUCTION

Undergraduate university students have substantial flexibility in choosing what courses they take and when they take them. Course sequencing is usually enforced only by a modest set of course prerequisites. This study examines the impact of different course sequences on student learning outcomes, as measured by course grades. The data used in this study includes eight years of undergraduate student grade data from Fordham University. Prior studies on course sequencing have generally been quite limited. Similar research has focused more on course selection, the optimal set of courses for a student to take to maximize performance or time to graduation [4, 5], than on course sequencing. Studies that focused on course sequencing were limited to a single discipline, such as communications [7] and psychology [2]. Our study considers all undergraduate courses within the university, including sequences that span disciplines. Prior studies also only considered how early courses predict performance in later courses, whereas our study does not have this restriction and focuses instead on maximizing overall student performance.

Our study considers the impact of sequencing on pairs of courses. This simplifies the analysis and reduces the risk of finding spurious correlations. The grade performance of students taking each pair of courses in the two possible sequential orderings is measured, with the goal of identifying the ordering that yields the best overall performance (concurrent registrations are excluded from the analysis). Comparing the grade performance for the two sequences required the development of new metrics, which we consider to be one of the contributions of this research. The methodology described in this study, along with the metrics that are introduced, are embodied in a publicly available software analysis tool [6].

Every possible course-pair sequence is considered as long as there are a sufficient number of students to provide reliable results. However, our analysis focuses primarily on course sequences within certain departments and groups of departments. This focus is due to our affiliation with a Computer Science department and the current focus on STEM (Science, Technology, Engineering, and Mathematics) education that is driven by national interests and the needs of industry. We also examine course pairs that include both humanities and STEM courses, because we are interested in the role that a liberal arts education has on STEM education.

There are many factors that can impact instructor performance [8], such as class size, course workload, and time of day of a class [1]. These factors also will impact student performance and hence can interfere with our ability to draw clear conclusions about course sequencing effects. In the present study, we normalize the grade data at the course section level to account for different instructor grading schemes, but do not address the other confounding factors. Our expectation is that the large number of course sections associated with most courses will limit the impact of these factors.

There are several uses for the course sequencing analysis described in this paper. The most obvious is that this information can be used to improve recommendations provided to students concerning beneficial course orderings. When these benefits are substantial enough, official course prerequisites can be modified. Beyond these direct applications of the work, the sequencing results can provide insight into the relationships between courses, and this can be used to inform academic policies. For example, if Course A is not generally considered relevant to Course B, but nonetheless leads to improved student performance in Course B, then one might want to recommend Course A to students who must take Course B.

2. METHODOLOGY

This section describes the data set used, the data preprocessing and transformation that is necessary to convert the data into a form suitable for analysis, and the evaluation metrics that measure the impact of course sequencing.

2.1 Initial Student Course-Grade Data Set

The initial data set describes the grade performance of each undergraduate student in all course sections with at least five students. Each of the 473,527 data set records, which collectively cover 24,969 distinct students, identify a student, a course

Tess Gutenbrunner, Daniel Leeds, Spencer Ross, Michael Riad-Zaky and Gary Weiss "Measuring the Academic Impact of Course Sequencing using Student Grade Data". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 799-803. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

(including the course section and semester), the instructor, and the student's grade in the course. Although we aggregate the information to course level, section information is used to normalize student grades. Unfortunately, the initial data set cannot be made publicly available due to strict student privacy laws.

2.2 Data Preprocessing and Transformation

The analysis conducted in this study is based on pairs of courses. From the initial student course-grade data set, we compute and maintain information for each course sequence A→B and B→A, where A and B represent arbitrary courses. For each of these sequences, we maintain a list of all students taking the two courses in the corresponding order, and the grades they receive in each course. The particular section each student enrolls in is also tracked, so that grades can subsequently be normalized at the section level. The transformation of the data from the student course-grade level to the course-pair sequence level, and the generation of our evaluation metrics, are accomplished using our publicly available Python-based tool [6].

In this study, a course pair is analyzed if it meets two conditions. The first condition ensures that the percentage of students taking the sequence in each direction exceeds *MinCSP*, the Minimum Course Sequence Percentage. For this study, *MinCSP* is set to 30%, which ensures that both orderings are taken at least 30% of the time. This excludes abnormal situations where a particular course sequence is rarely taken, such as when a student takes an introductory class in their senior year or retakes a failed course outside of the normal order. The second condition ensures that at least a minimum number of students, *MinCount*, aggregated over all course sections, takes the courses in each order. *MinCount* is utilized to ensure that the sample size is sufficient to generate reliable results. For this study *MinCount* is set to 50 students.

Table 1 specifies how many course pairs remain after these conditions are applied. The conditions are applied sequentially, with *MinCSP* applied before *MinCount*. The values in the rightmost column reflect the number of course pairs actually analyzed. Table 1 displays the number of course pairs for the entire data set, as well as for the five course subsets that are of particular interest to us. Our university has no engineering school, so the STEM courses are offered by the Biology, Chemistry, Computer Science, Mathematics, Natural Sciences, Physics, and Psychology departments. The Humanities courses include all courses from the African and African American Studies, Anthropology, Art History, English, Philosophy, Theology, and Visual Arts departments.

Table 1. Number of course pairs for different course subsets

Data Set	Threshold		
	None	MinCSP=30%	MinCount=50
Full Data Set	81,327	21,461	1,939
Computer Science	850	253	14
Mathematics	392	92	23
Mathematics and CS	1,724	490	51
STEM	12,055	3,000	291
STEM & Humanities	27,303	6,646	684

2.3 Evaluation Metrics

Several metrics are used to analyze the impact of course sequencing on student performance. These metrics are based on lower-level metrics, which are introduced first. Ultimately, we want to see how the mean grades for each course in a course pair are impacted by course order in order to determine the optimal ordering and net benefit in grade performance.

The first step computes the mean grades for each course in a course pair for each of the two orderings. Because instructors vary widely in their leniency when assigning grades, all grades are normalized at the course section level using z-score normalization, as described by Equation 1. In this equation x_i represents the grade of student *i* in the course section, μ represents the mean section grade over x_i, and σ represents the standard deviation of the section grades.

$$Z_i = (x_i - \mu) / \sigma \quad (1)$$

For every course pair <*A*, *B*> we determine the average normalized grade for each course based on each ordering. Specifically, we compute $\mu_A(B \rightarrow A)$, $\mu_A(A \rightarrow B)$, $\mu_B(A \rightarrow B)$, and $\mu_B(B \rightarrow A)$, where the subscript of μ denotes the course for which the normalized mean is computed and A → B indicates that course A is taken before course B (and vice versa for B → A). As an example, for the course pair <*Math I*, *English I*>, $\mu_{Math\ I}$(*English I*→*Math I*) represents the mean normalized grade in *Math I* for students who took *Math I* after *English I*.

These normalized means are used to compute the difference in mean normalized grades (DNG). Two DNG values are computed for each course pair <*A*, *B*> since the difference in normalized mean grades is computed for each course. Equations 2 and 3 define these values, where $DNG_{A:B}$ is the difference in mean normalized grade for Course A when Course A is taken after course B rather than before course B, and $DNG_{B:A}$ is the difference in mean normalized grades for Course B when Course B is taken after course A rather than before course A. We compute the difference using the order noted in the equations, because we generally expect a course to perform better when it is taken second and anticipate that most DNG values will be positive.

$$DNG_{A:B} = \mu_A(B \rightarrow A) - \mu_A(A \rightarrow B) \quad (2)$$

$$DNG_{B:A} = \mu_B(A \rightarrow B) - \mu_B(B \rightarrow A) \quad (3)$$

The DNG equations measure the benefit of taking two courses in a particular order, but do not reflect the net benefit of one ordering over the other (if both DNG values are positive then the difference between the orderings will be reduced). We therefore compute the order benefit, OB, which is the net difference in DNG values of one ordering over the other. The OB is defined relative to a specific course ordering, as indicated in Equation 4. The OB value will be calculated for both possible orderings, but we will only list the one that is positive, which indicates the optimal course ordering.

$$OB_{A \rightarrow B} = DNG_{B:A} - DNG_{A:B} \quad (4)$$

We work through an example using <*Math I*, *English I*>, assuming the following statistics:

$\mu_{Math\ I}(English\ I \rightarrow Math\ I)$ = 0.40
$\mu_{Math\ I}(Math\ I \rightarrow English\ I)$ = -0.05
$\mu_{English\ I}(Math\ I \rightarrow English\ I)$ = 0.40
$\mu_{English\ I}(English\ I \rightarrow Math\ I)$ = -0.10

Assuming *Math I* takes on the role of Course A and *English I* Course B, using Equation 2, $DNG_{A:B}$ = 0.40 – (-0.05) = 0.45, and using Equation 3, $DNG_{B:A}$ = 0.40 – (-0.10) = 0.50. Applying Equation 4, we get $OB_{A \rightarrow B}$ = 0.50 – 0.45 = 0.05. These results are summarized in the first row of Table 2. The assignment of the two courses to A and B is arbitrary, so we can reverse them, which corresponds to the course ordering in the second row of Table 2. Then, using Equation 2 and Equation 3, we get $DNG_{A:B}$ = 0.50 and $DNG_{B:A}$= 0.45, which yields an OB value of 0.45 – 50 = -0.05. The values of $DNG_{A:B}$ and $DNG_{B:A}$ in Table 2 are flipped when we

reverse the roles of A and B (compare rows 1 and 2). This is logically and mathematically required given the definition of the DNG metric, so the OB value of one ordering must equal the negative of the other. The results in Table 2 show that taking *Math I* and then *English I* yields an overall improvement in normalized grades of 0.05, whereas taking the courses in the reverse order yields a net deterioration of 0.05.

Table 2. Example of a course pairing

Course A	Course B	$DNG_{A:B}$	$DNG_{B:A}$	$OB_{A \to B}$
Math I	English I	0.45	0.50	0.05
English I	Math I	0.50	0.45	-0.05

3. RESULTS

This section provides selected results from our analysis, with a focus on the difference in normalized grades for different course sequences. Order benefit is our primary metric, as it summarizes the net benefit of a particular course sequence over the alternative, but DNG is also informative since it specifies the amount of benefit in taking one course before the other. For example, it is possible for two competing sequences to have identical positive DNGs, leading to a zero order benefit. Top order benefit results are presented for course sequences restricted to: Computer Science, Math, Math and Computer Science, STEM, STEM and Humanities, and "All Courses" across all disciplines. We posit explanations for some of the results based on our knowledge of the domain.

The top three order benefit values for computer science courses are displayed in Table 3. Note that while the sequence *Computer Algorithms* → *Data Mining* has the highest OB value, based on the $DNG_{B:A}$ values, taking *Data Communications and Networks* after *Data Mining* yields a slightly greater improvement than taking *Data Mining* after *Computer Algorithms*. The key difference is that taking each of those pairs of courses in the opposite order (i.e., $DNG_{A:B}$) yields very different results. The two negative $DNG_{A:B}$ values in Table 3 indicate that the corresponding courses yield worse results when they are taken second. Specifically, students in *Computer Algorithms* perform worse when they take it second. We generally would not expect this to occur. This result may stem from weaker students who delay taking *Computer Algorithms*.

Table 3. Computer Science courses with largest order benefit

Course A	Course B	$DNG_{A:B}$	$DNG_{B:A}$	OB
Computer Alg.	Data Mining	-0.110	0.233	0.343
Data Structures	Computer Organization	-0.073	0.103	0.176
Data Mining	Data Comm. & Netwks.	0.101	0.235	0.134

A plausible explanation for the first entry in Table 3 is that *Data Mining* utilizes some knowledge of *Computer Algorithms* and hence taking *Data Mining* second is beneficial. While the same reasoning could be applied to the reverse ordering, the negative DNG indicates no benefit for that ordering, possibly because *Data Mining* does not teach the basics of computer algorithms. With respect to the entry in the second row of Table 3, the benefit of foundational mathematics and algorithmic knowledge provided by *Data Structures* is apparent in the somewhat more application-oriented *Computer Organization* course.

Table 3 shows negative $DNG_{A:B}$ values are smaller in magnitude than positive $DNG_{B:A}$ values — a finding replicated in subsequent tables. The presence of negative $DNG_{A:B}$ values may be an artifact of our focus on course pairs with the highest overall order benefit, because order benefit is maximized when $DNG_{A:B}$ is negative.

Table 4 shows the results for three sequences of mathematics courses. The third entry is the easiest to explain. *Business Finite Math* and *Finite Math* cover similar material, but the former covers more basic material. Students are not generally expected to take both courses, but if they do, they most likely will take the more basic one first. *Discrete Math* provides a background in formal proofs, which appears to benefit from advanced mathematical experience (*Multivariable Calculus I*) and to provide benefit to advanced study of calculus (*Multivariable Calculus II*).

Table 4. Mathematics courses with largest order benefit

Course A	Course B	$DNG_{A:B}$	$DNG_{B:A}$	OB
Discrete Math	Multivar. Calc II	-0.056	0.252	0.308
Multivar. Calc. I	Discrete Math	-0.041	0.249	0.290
Business Finite Math	Finite Math	-0.024	0.145	0.169

Most computer science programs require several mathematics courses, but the specific impact of the math courses on computer science courses is not well understood. Table 5 explores the relation between the two departments, restricting the sequences to include one math course and computer science course. One of the more notable results is the entry in the first row. Both courses teach finite mathematics, but *Structures of Computer Science* is offered by the Computer Science department and is intended for non-majors, while *Finite Math* is offered by the Mathematics department. *Structures of Computer Science* also devotes several weeks to cover simple programming assignments, thereby further reducing the time spent on the mathematics content. For these reasons, it is reasonable to conclude that the sequence with the high OB value corresponds to taking the more basic course first. It is also noteworthy that *Calculus I* has a very positive impact on taking programming courses (*Computer Science I* and its lab) and *Structures of Computer Science*. Thus it appears that increased mathematical sophistication does have a positive impact on computer science and computer programming. This is especially interesting because the mathematical material in *Calculus I* has only a tangential relationship with computer science. Most computer science programs require calculus, and our empirical data justifies this requirement.

Table 5. Math and CS courses with largest order benefit

Course A	Course B	$DNG_{A:B}$	$DNG_{B:A}$	OB
Structures of CS	Finite Math	-0.002	0.429	0.431
Calculus I	CS I	-0.035	0.338	0.373
Calculus I	CS I Lab	-0.012	0.252	0.264
Calculus I	Structures of CS	-0.010	0.213	0.223

Table 6 displays the remaining results for the three groupings of sequences: STEM courses, mixed STEM and humanities courses, and all courses without any restrictions. The first entry under the STEM category shows a benefit in taking *Applied Calculus I* after *General Chemistry I*. This ordering is typical for students on the Pre-Health track who wish to go to medical school, which may explain the high order benefit, since these students are generally motivated to achieve high grades. Furthermore, under the STEM category we find a benefit for *Learning (Psychology)* followed by *Multicultural Psychology*. The first psychology course in this sequence is a 2000 level course while the second is a 3000 level course, indicating yet again that there is a benefit from taking a more advanced course in the same discipline second.

Looking at the STEM & Humanities courses, students who took *Organic Chemistry I→ Intro. to Cultural Anthropology* did significantly better in both classes, as demonstrated by the magnitudes of the DNG values (the negative $DNG_{A:B}$ indicates *Organic Chemistry I* does worse when taken second and hence performs better when taken first). The same pattern is replicated with an even higher OB when considering the *Organic Chemistry Lab*. Pre-Health students tend to take *Organic Chemistry* very early in their college career and may dominate that particular course ordering.

The first row under the "All Courses" category displays the sequence *Spanish Language & Literature → Christian Hymns* with a very high order benefit. Students performed best in each of the two courses when taking them in the specified sequence. This may be due to the fact that Spanish literature is heavily influenced by Christianity, and therefore provides important background for students who plan to take *Christian Hymns*. Explanations for the other entries may require consultation with faculty from the associated departments.

Table 6. STEM, STEM/Humanities, All courses with large OB

Course A	Course B	$DNG_{A:B}$	$DNG_{B:A}$	OB
STEM Courses				
General Chem. I	Applied Calculus I	-0.17	0.400	0.570
Intro. Astronomy	Abnormal Psych.	-0.187	0.309	0.496
Learning (Psych.)	Multicultural Psych.	-0.021	0.419	0.440
Intro. Bio. I	Structures of CS	-0.152	0.283	0.435
Structures of CS	Finite Math	-0.002	0.429	0.431
Gen. Chem. Lab I	Structures of CS	-0.102	0.325	0.427
Calculus I	CS I	-0.035	0.338	0.373
Intro. Bio Lab I	Structures of CS	-0.069	0.297	0.366
Physics II Lab	Human Physiol. Lab	-0.019	0.287	0.306
STEM & Humanities Courses				
Org. Chem. Lab I	Intro. Cultural Anthr.	-0.520	0.606	1.126
Organic Chem. I	Intro. Cultural Anthr.	-0.330	0.554	0.884
Forensic Science	Philosophical Ethics	-0.310	0.474	0.784
Texts & Contexts	Discrete Math	-0.234	0.372	0.606
All Courses				
Spanish Lang. & Lit.	Christian Hymns	-0.436	0.714	1.150
Medieval History	Intro. Media Industry	-0.178	0.550	0.728
Composition II	Intro. Archaeology	-0.218	0.494	0.712
Sociology Focus	Faith & Crit. Reason	-0.134	0.565	0.699
Calculus II	Intro Sociology	-0.111	0.488	0.599
American History	Personality (Psych)	-0.095	0.487	0.582

4. CONCLUSION

The research described in this study introduced a methodology and set of metrics for assessing the impact of course sequencing on student performance. The analysis of our results focuses on several disciplines, such as Computer Science and Mathematics, as well as higher level groupings, such as STEM courses. Many of the results demonstrate that there is a substantial benefit with a particular sequencing of courses, such as taking *Finite Math* after *Structures of Computer Science* or taking *Computer Science I* after *Calculus I*. Our methodology and metrics are implemented in our Python-based software tool [6], which can be used by other researchers.

The course sequencing results in this paper can be used to assist with course recommendations and can be used to inform, and even modify, course prerequisites. For example, our results show a larger than expected benefit of taking calculus before a programming course; additional analysis and data will be needed to see if this extends to a broader set of mathematics courses, but if it does, then new prerequisites perhaps should be added. The results in this study also provide insight into the inter-relationships between courses and disciplines.

Many of our observed results can be explained based on our knowledge about college education and domain knowledge of specific disciplines. However, in some cases explanations are not readily available. Our search for explanations of why one sequence may outperform another can also benefit from additional domain knowledge, as our knowledge is mainly limited to computer science. Course syllabi could also prove to be useful. It would also be very interesting to apply our methodology to data from different universities, and we hope to do this in the future. It would be informative to see if the course sequencing patterns present in our university hold elsewhere. Although our university is relatively large, in many cases the number of students taking some pairs of courses was relatively small, and this informed our relatively low *MinCount* threshold of 50. With more data, we could increase this threshold, which would diminish the impact of factors like instructor effectiveness.

Our methodology normalizes for some external factors, such as different instructor grading schemes, but does not account for all factors that can impact student performance. In particular, we suspect that some course sequencing results are due to certain populations of students (e.g., Pre-Health students) taking courses in one particular order over another. In future work we do plan to consider some of these factors and modify our evaluation to isolate their impact. In cases where that is not feasible, we will at least provide summary statistics to assess the influence of these factors. For example, since we suspect that academically stronger students sometimes take courses in a different sequence than weaker or less motivated students, we can compare the overall GPAs of students taking the courses in each course ordering and note when they exhibit a statistically significant difference. Alternatively, we can normalize for overall student GPA, something that we are currently doing in a study on instructor effectiveness.

One final area that we plan to pursue is better evaluation of our results. One way to do this is to utilize statistical significance testing. Given the number of potential patterns we can find with the large number of pairs of courses, we may need to set our p-value quite low. We may be able to improve this situation by limiting our course interactions to courses within a single department or between related fields (e.g., Biology and Chemistry). We can also validate our results by partitioning the data into a training and test set and subsequently verifying if the patterns found in the training data hold for the test data. In this regard, the differences in student performance can be viewed as predictions, so the standard training and test set evaluation methodology applies.

The data utilized in this study is itself a valuable resource. Our research group has analyzed this data in a variety of ways to provide additional insights. Two studies have used this data to group/cluster courses and analyze the interrelationships between courses. One of these studies uses course co-enrollments to form the clusters and to identify hub courses [9], while the other uses the correlation between students grades as a similarity metric to cluster the courses [3]. Both of these studies used their respective notions of similarity to form networks of courses, and then analyzed these with existing network analysis techniques.

5. REFERENCES

[1] Anderson, C. J., Poulsen, S., and West, M. 2020. The relationship between course scheduling and student performance. In *Proceedings of the 4th Educational Data Mining in Computer Science Education Workshop* (collocated with EDM2020), July 10, 2020.

[2] Betancur, L., Rottman, B. M., Votruba-Drzal, E., and Schunn, C. 2019. Analytical assessment of course sequencing: The case of methodological courses in psychology. *Journal of Educational Psychology*, 111(1), 91.

[3] Leeds, D. D., Zhang, T., and Weiss, G. M. 2021. Mining course groupings using academic performance. In *Proceedings of the 14th International Conference on Educational Data Mining*.

[4] Morsy. S., and Karypis, G. 2019. Will this course increase or decrease your GPA? Towards grade-aware course recommendation. *Journal of Educational Data Mining*, 11(2):20–46, 2019.

[5] Parks, M. R., Faw, M., & Goldsmith, D. 2011. Undergraduate instruction in empirical research methods in communication: Assessment and recommendations. *Communication Education*, 60, 406-421. DOI= 10.1080/03634523.2011.562909

[6] Riad-Zaky, M., Weiss, G.M., and Leeds, D.D. Course Grade Analytics with Networks (CGAN) [computer software], *Available: http://www.cis.fordham.edu/edmlab/software*

[7] Richards, A. S. 2012. Course sequencing in the communication curriculum: A case study. *Communication Education*, 61(4), 395-427. DOI= 10.1080/03634523.2012.713500

[8] Wachtel, H. K. 1998. Student Evaluation of College Teaching Effectiveness: a brief review, *Assessment & Evaluation in Higher Education*, 23:2, 191-212, DOI= 10.1080/0260293980230207

[9] Weiss, G.M., Nguyen, N., Dominguez, K., and Leeds, D.D. 2021. Identifying hubs in undergraduate course networks based on scaled co-enrollments. In *Proceedings of the 14th International Conference on Educational Data Mining*.

Mining Course Groupings using Academic Performance

Daniel D. Leeds, Tianyi Zhang, Gary M. Weiss
Department of Computer and Information Science
Fordham University, New York, NY
{dleeds, tzhang130, gaweiss}@fordham.edu

ABSTRACT

This study computes the correlation of student grades between pairs of courses in a large university. Course network graphs are then generated, where courses are represented as nodes and courses are connected if they have a high degree of grade correlation. Graph mining and network analysis tools visualize the course networks, identify course clusters and course cliques, and compute informative network statistics. Results are analyzed for pairs of courses and courses grouped by academic department or program of study. Strong course similarity groupings are observed within scientific disciplines, between pre-health courses, and within subfields of computer science. No prior study using this notion of course similarity has been conducted.

Keywords

Educational, Clustering, Correlation, Network graphs

1. INTRODUCTION

This paper describes a method for grouping and analyzing courses based on similar student performance, where similarity is measured between pairs of courses using the Pearson correlation of the grades assigned to students who take both courses. A graph is then formed that represents courses as nodes, and has edges between course-pairs when the student grade correlation is above a specified threshold. The resulting graph is then analyzed using a variety of graph analysis techniques, to provide insights into the relationship between individual courses and course groupings. Data preprocessing steps are described to handle confounding factors, such as differing instructor grading schemes. The methodology is encapsulated in a software tool that was developed for this study and is publicly available [5]. This study utilizes eight years of undergraduate student course grade data from Fordham University. The results show that there are strong connections between pre-health courses and courses within subdisciplines of computer science, and that courses that teach specific skills are much more highly connected to other courses than introductory survey courses.

Daniel Leeds, Tianyi Zhang and Gary Weiss "Mining Course Groupings using on Academic Performance". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 804-808. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

The knowledge gleaned from this research can be used to influence curriculum design and academic policies. For example, if a student performs poorly in the first course within a set of highly correlated courses, then they are likely to encounter future difficulty; therefore, they could be asked to repeat the course or be offered academic assistance. Results from this study have many possible applications, but as is the case with descriptive data mining tasks, it may take some time to discover some of them. However, we feel that the course correlation networks that we generate and the various metrics that we introduce are themselves key contributions, which will lead to further research in educational data mining. This study is unique in that no other analysis of university courses is based on a notion of similarity that relies exclusively on student performance. One study, which is superficially similar, measures course similarity based on student course co-enrollments [7]. That study, also conducted by our research group and based on the same data set, uses this much more traditional notion of similarity to perform similar analyses; namely course network graphs are generated and then analyzes using existing network analysis methods and metrics.

2. DATASET DESCRIPTION

Eight years of student-course records were obtained from three of Fordham university's undergraduate colleges, where each record describes the performance of a student in a course section. This study restricts the data and analysis to: pre-health courses required for medical school admission, popular university core curriculum courses, and Computer Science and Psychology courses (a detailed analysis would not be possible if courses from all 83 majors were included). Computer Science and Psychology courses were included due to our affiliations with those departments, while core curriculum courses were chosen because of their prominence in our university and their diversity (students complete more than twenty core courses covering philosophy, history, foreign languages, performing arts, mathematics, and science). Pre-health courses are included because they cover many key introductory STEM courses. This study will be expanded to other disciplines in the future.

Table 1 summarizes the data and its distribution across the course categories. The core courses contribute more than half of the total course sections and are largely responsible for the data covering $20,797$ students. Each record corresponds to one student in one course section and includes the following features: *student ID*, *final grade*, *department name*, *course number*, *course title*, *semester*, and *section number*.

Table 1: Distribution across Course Categories

Course Category	Records	Sections	Courses
Computer Science	14,137(13%)	705(15%)	53(39%)
Psychology	18,017(17%)	966(20%)	67(50%)
Core	62,005(58%)	2,706(56%)	8(6%)
Pre-Health	13,087(12%)	434(9%)	7(5%)
Total	107,246	4,811	135

The final grade uses a 4 point scale and most courses will have many sections. Student privacy concerns prohibit us from sharing the raw data, even though the student identifier values have been anonymized; however, the course correlation matrix central to our analysis is available [6].

3. DATA PROCESSING

An overview of the process for measuring similarity between courses is provided in Section 3.1, and the individual steps are described in successive subsections. The code that implements these steps is publicly available [5].

3.1 Overview

The initial data set contains records that describe the performance of each student in a each course section. A variety of preprocessing steps are executed, as summarized in Fig. 1. A course correlation matrix that measures the similarity of each pair of courses using the Pearson correlation of student grades is generated in Step 7. The course network graphs, modularity clusters, and course cliques are then generated from this correlation matrix, as described in Section 4.

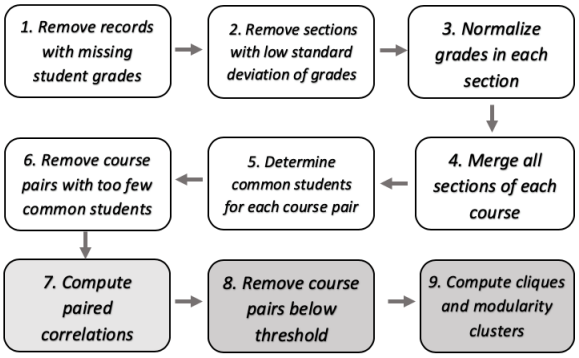


Figure 1: Overview of data processing steps

3.2 Initial Data Cleaning (Steps 1 and 2)

The first step removes records that do not have numerical grades, such as courses taken pass/fail. Some instructors sometimes assign students very similar grades, which makes it difficult to assess the similarity of courses based on grades. For this reason, Step 2 removes course sections where the standard deviation (σ) of student grades is below a specified threshold. This requires aggregation of the student course records to the section level, which yields $4,811$ sections. Fig. 2 provides the distribution of standard deviation values across these sections, and also provides a curve that shows the number of records and percentages of sections that are kept for each standard deviation threshold value (for each value we discard the sections with a lower threshold). Based on Fig. 2 we consider the values of 0.20, 0.30, and 0.40 to be reasonable candidates that maintain the majority of course sections. We ultimately selected a threshold of 0.30, which drops 6% of the sections and eliminates 6 courses (which are not left with any sections).

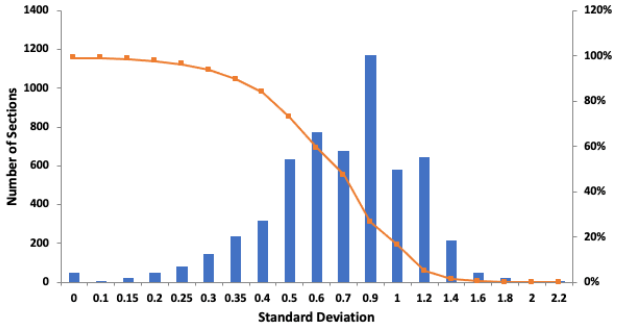


Figure 2: Distribution of grade standard deviation

3.3 Grade Normalization (Step 3)

Instructors may be easy or hard graders, and these differences will cause problems with grade correlation when a course is taught by multiple instructors. This issue is remedied by applying z-score normalization to the grades in each course section, which substracts the mean section grade from each grade and then divides it by the standard deviation of the section grades.

3.4 Generate Course-Pair Grades (Step 4-6)

Step 4 aggregates the data from the section level to the course level, which may combine dozens of course sections, spanning many years. Step 5 then forms pairs of courses, keeping on the grade data from students common to both courses. Course pairs are formed from every course that remains after application of the $\sigma = 0.3$ threshold in step 2. Step 6 then filters the course pairs that do not have at least 20 students in common, to ensure that the grade correlation is meaningful. This results in the removal of $4,585$ (25%) of the remaining course pairs.

3.5 Compute Paired Correlations (Step 7)

The final preprocessing step computes the Pearson correlation [2] between the remaining course pairs, which generates the correlation matrix that is central to our analysis. A small sample of the correlation matrix is provided in Table 2. The complete correlation matrix is publicly available [6]. Entries in the correlation matrix are not impacted by order, so values above the diagonal are omitted. Null values occur when a course pair does not have enough common students. In Table 2 we see that, as expected, there is a high correlation (0.94) between *Discrete Structures* and the associated lab. There is also a strong correlation (0.81) between *Computational Neuroscience* and *General Physics I*, which may be due to the heavy use of mathematical modeling of physical systems in both classes. *Bioinformatics* and *General Physics I* exhibit a low correlation (0.19), perhaps reflecting a heavier practical programming focus in the bioinformatics course. It is surprising that *Discrete Structures* and *Computer Algorithms* have a relatively low correlation (0.37), since they both require similar mathematical reasoning skills. This suggests that the *Discrete Structures* may not be preparing students sufficiently for future coursework.

Table 2: Representative Course-Pair Correlations

	Disc Struct	Disc Lab	Web Prog	Comp Neuro	Comp Alg	Bioinf	Gen Phys-I
Disc Struct	1						
Disc Lab	0.94	1					
Web Prog	–	–	1				
Comp Neuro	–	–	–	1			
Comp Algs	0.37	0.33	0.41	–	1		
Bioinfor	–	–	0.79	0.47	0.24	1	
Gen Phys I	–	–	–	0.81	–	0.19	1

4. RESULTS

This section describes the results derived from the course-pair correlation matrix. Section 4.1 covers the correlation results between individual course pairs, Section 4.2 covers the cliques within the course correlation graph, and Section 4.3 analyzes the course correlation network graphs.

4.1 Analysis of Course-Correlation Pairs

The distribution of Pearson course-pair correlations is displayed in Fig. 3. The leftmost bar is due to correlations between a course and itself. The top 25% of course-pair have a correlation greater than 0.5. The course network correlation graphs in Section 4.3 are generated using a threshold of 0.5.

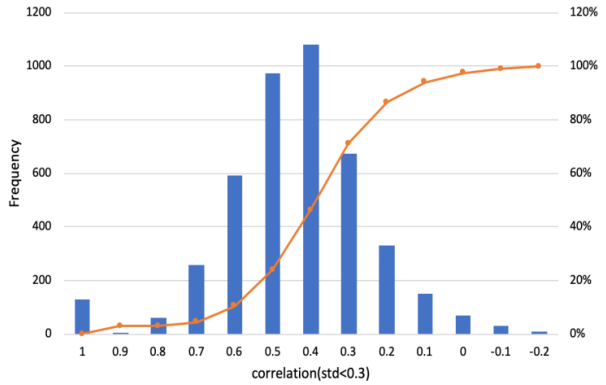


Figure 3: Distribution of course-pair correlations

Table 3 lists course pairs with correlations > 0.75. The top three entries cover matching lecture and lab courses, which is unsurprising since they cover complementary material. More than 80% of the entries are contained within an academic department, although there are interesting inter-departmental entries. The link between *General Physics I* and *Computational Neuroscience* was previously discussed and involves mathematical modeling. The link between *General Chemistry Lab II* and *Computer Algorithms* is not obvious, but both involve designing and applying a precise sequence of instructions. *Philosophy of Human Nature* shows an interesting connection with *Infant and Child Development*, potentially establishing a link between Philosophy and Psychology. The Philosophy class's link to *Scientific Computing* is more difficult to explain, although it may be related to the interdisciplinary nature of *Scientific Computing*.

4.2 Clique Results

A k-clique is a set of k nodes that are each directly connected to each other by an edge. Table 4 shows the number of cliques of each size in the course correlation network graph for correlation thresholds (ρ) of 0.55, 0.55 and 0.6. The table

Table 3: High Correlation (ρ) Course-Pairs

Course 1	Course 2	ρ
Discrete Struct II	Discrete Struct II Lab	0.96
Comp Sci II	Comp Sci II Lab	0.95
Comp Sci I	Comp Sci I Lab	0.93
Gen Phys I	Comp Neuro	0.81
Intro Bio I	Intro Bio Lab I	0.79
Web Program	Bioinformatics	0.79
Learning	Health Psychology	0.78
Perception Lab	Law and Psychology	0.78
Gen Chem Lab II	Comp Algorithms	0.78
Phil of Human Nature	Infant & Child Devel	0.78
Phil of Human Nature	Scientific Computing	0.77
Psych & Human Vals	Research Methds Lab	0.77
Law and Psych	Clinical Child Psych	0.77
Biopsych	Sens & Percep Lab	0.76
Intro Robotics	DataComm & Networks	0.76

shows that increasing the correlation threshold even slightly dramatically reduces the number of cliques, and hence we use 0.5 to retain a clear picture of course network structure. Each clique has many sub-cliques (e.g., each 7-clique has 7 6-cliques and 21 5-cliques), which we view as redundant, and hence the table excludes all sub-cliques. Cliques may span different course categories or fall entirely within one category. Table 5 shows how the cliques from Table 4 are distributed across the five course categories using $\rho = 0.5$. Cliques that do not fall within one category are included in the "Span" field.

Table 4: Number of Cliques as ρ Threshold Varies

Clique Size	$\rho \geq 0.5$	$\rho \geq 0.55$	$\rho \geq 0.6$
3-cliques	172	66	29
4-cliques	51	50	4
5-cliques	56	2	0
6-cliques	15	0	0
7-cliques	4	0	0
8-cliques	1	0	0

Table 5: Number of Cliques in Each Category

Clique Size	CS	Psych	Core	Pre-H	Span
3-cliques	46	9	0	0	117
4-cliques	11	32	0	0	8
5-cliques	14	39	0	0	3
6-cliques	0	15	0	0	0
7-cliques	0	3	0	1	0
8-cliques	0	1	0	0	0

Psychology courses form most of the large cliques with size 6 and greater. Psychology courses are more grouped together than Computer Science courses, which have many smaller-sized cliques. The 7 pre-health courses form a single clique, which suggests that performance in these courses is based on similar abilities or knowledge. Core courses lack even smaller 3 cliques. Despite their shared mission of core liberal arts training, it appears the differences in subject matter prevents similarity in course performance. No large cliques span course categories, but when $k = 3$, spanning cliques outnumber the other ones, which suggests that cliques only become meaningful at larger sizes. The largest cliques associated with the Computer Science, Psychology, and Pre-health courses are described in Table 6 of the appendix. Most of those cliques cover related courses (e.g., a 5-clique in Computer Science covers programming courses).

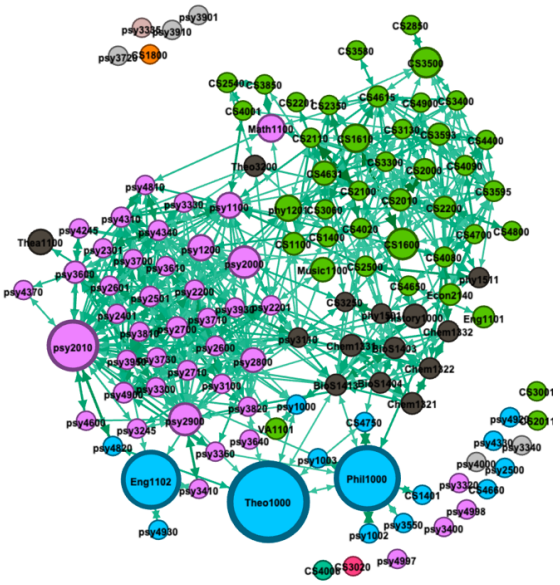


Figure 4: Network graph (all categories).

4.3 Course Correlation Network Graphs

The course correlation graphs generated with $\rho = 0.5$ were supplied to the Gephi social network analysis software [1]. Gephi partitions highly connnected nodes into modularity classes and assigns each a different color [3]. The size of each node is determined by ranking the node's "betweenness centrality," which is based on how often a node appears on shortest paths between all nodes in the network [4].

Fig. 4 shows the Gephi network that includes all courses. Nodes are labeled with a department abbreviation ("Eng" for English and "CS" for Computer Science), and 4-digit course number. Course numbers are not informative so our analysis refers to courses by title as needed. The figure shows a clear partitioning of courses between Computer Science (green, right) and Psychology (purple, left), with Pre-health courses (dark grey and below Computer Science) clustered together and forming a partial bridge between Computer Science and Psychology. While individual edges are difficult to distinguish, the figure shows that courses within a category are much better connected to each other than to courses in other categories. First-year core curriculum courses *English 1102*, *Theology 1000*, and *Philosophy 1000* are very large, indicating their large betweenness-centrality. These courses therefore often occur in the shortest paths between other courses and act as bridges between parts of the network. While these core courses do not have many connections, they connect to a diverse set of courses. *Philosophy 1000* is connected to well-connected courses from Economics, Psychology, and Computer Science, while. *Theology 1000* is connected to classes in Psychology, Pre-health (Biology), and *Philosophy 1000*. These core classes appear to be an indirect indicator of performance for classes across the university. Both classes introduce and carefully study selected core concepts in their respective fields.

Network graphs focusing on Computer Science courses and Psychology courses are provided in the appendix in Fig. 5 and Fig. 6, respectively. The modularity classes in Fig. 5 correspond to meaningful subdisciplines of Computer Science: the light-blue modularity class covers Information Science courses like Data Mining (4631); the magenta modularity class covers programming courses such as CS1 and Lab (1600, 1610), CS2 and Lab (2000, 2010), UNIX programming (3130), and Scientific Computing (4750); and the orange modularity class covers advanced courses like Algorithms (4080), Theory of Computation (4090), and Operating Systems (3595). The modularity class groupings differ from the cliques in Table 6 of the appendix, although both group the same programming courses together. Furthermore, the five largest nodes in the Fig. 5 based on betweeness centrality (*4631 Data Mining*, *4615 Data Communications*, *3593 Computer Organization*, *2200 Data Structures*, and *3300 Web Programming*) are well represented in the Computer Science cliques in Table 6. For Computer Science, high betweenness centrality reflects an abundance of both one-step and few-step connections to other courses. Within the department, it is known that a student with a poor grade in one of these classes will often struggle in the major. Most of these classes are designed to hone specialized skills within Computer Science. The key observation from the Gephi graph of Psychology courses in Fig. 6 is that the Research Methods Lab course is strongly connected with other psychology courses, while the introductory survey course is very poorly connected. This suggest that classes focused on specialized skills are more predictive of performance in advanced classes than a general survey class.

5. CONCLUSION

This descriptive data mining study defined an innovative notion of course similarity based on student performance, and then used this similarity metric to form course network graphs. These network graphs were then used to analyze the relationship between courses and course groupings. This methodology was applied to eight years of undergraduate student data at a large university.

The study established that there are many course pairs for which student performance is highly correlated. When requiring at least 20 common students, 25% of course pairs exceed the 0.5 correlation threshold used in this study, and 5% of pairs exceed 0.7 correlation. Courses with the highest correlations are often offered by the same department. In addition, multi-course clusters naturally occur, especially within subdisciplines of an academic department, such as the programming courses within Computer Science. Course clusters were identified as cliques and modularity classes within the course correlation networks. As an extreme example, all pre-health courses formed a single clique. A small number of courses with high betweeness centrality were shown to link a diverse set of topics—within one discipline or between disciplines, and those courses connecting discplines were much more likely to introduce specific skills than to provide a broad survey of an area.

This paper also introduced a methodology for generating a course grade correlation matrix from student data, and included several steps to address confounding factors such as differing instructor grading policies. This methodology is available to other education researchers through our software and associated documentation [5]. Our work presented a new way of looking at course relationships by a novel way of measuring similarity. We plan to continue to investigate this notion of course similarity and to apply it to a larger set of courses.

6. REFERENCES

[1] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 3, 2009.

[2] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[4] U. Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[5] M. Riad-Zaky, G. M. Weiss, and D. D. Leeds. *Course Grade Analytics with Networks (CGAN) [computer software]*, April 2021. https://www.cis.fordham.edu/edmlab/software.

[6] G. M. Weiss and D. D. Leeds. *Fordham University Course Correlation Matrix [data file]*, January 2021. https://www.cis.fordham.edu/edmlab/datasets.

[7] G. M. Weiss, N. Nguyen, K. Dominguez, and D. D. Leeds. Identifying hubs in undergraduate course networks based on scaled co-enrollments. In *Proceedings of the 14th International Conference on Educational Data Mining (EDM 2021)*, 2021.

APPENDIX

Table 6 lists the large cliques associated with Computer Science, Psychology, and Pre-health courses. Many of the cliques have a common theme. Computer Science's second 5-clique includes three internet-focused courses: *Web Programming*, *Client Server Computing*, and *Data Communications*, while the third clique is dominated by programming courses (*Operating Systems* is an exception but includes programming projects). Psychology's 7-clique links classes covering complementary and overlapping elements of cognition; however, the 8-clique appears to span diverse topics. As mentioned earlier, the pre-health clique covers core science courses required by medical schools.

Table 6: Large Cliques in Different Categories

COMPUTER SCIENCE		
5-Clique	**5-Clique**	**5-Clique**
Data Mining	Data Mining	Comp Sci II
Web Programming	Web Programming	Comp Sci II Lab
Data Struct.	Data Comm.	Data Struct.
Client-server Comp	Client-server Comp	Operating Systems
Comp. Org.	Comp. Org.	Scientific Comput.
PSYCHOLOGY		
8-Clique		
Child Develop.	Biopsy.	Research Methods
Learning	Social Psych Lab	Human Sexuality
Aging and Society	Law and Psych	
7-Clique		
Child Develop.	Personality	Abnormal Psych
Intro Clin. Psych	Found. of Psych	Social Psych
Cognitive Psych		
PRE-HEALTH		
7-Clique		
Intro Bio I	Intro Bio II	Intro Bio Lab I
Gen Chem I	Gen Chem II	Gen Chem Lab I
Gen Chem Lab II		

The Gephi course correlation network graph for the Computer Science is displayed in Fig. 5. The contents of Fig. 5 were describe in detail in Section 4.3 and highlighted how the different modularity classes correspond to different subdisciplines within computer science. The Gephi course correlation network graph for Psychology, which was only briefly described in Section 4.3, is displayed in Fig. 6. Meaningful subcategories are much harder to identify, but it is notable that Research Methods Lab (2010) is most strongly connected with other psychology courses, indicating a valuable skill shared across the category. This contrasts with the required introductory survey class, Psychology 1200, which has a much lower betweenness centrality. This indicates that a class focused on specialized skills is more predictive of performance in more advanced classes than a general overview class. The psychology courses with largest betweenness centrality are all represented in the cliques in Table 6. The top four courses based on betweenness centrality are: *2010 Research Methods*, *2900 Abnormal Psychology*, *2800 Personality*, and *2700 Child Development* – all courses with specialized foci. As in Computer Science, high betweenness centrality in Psychology reflects an abundance of both one-step and few-step connections to other courses.

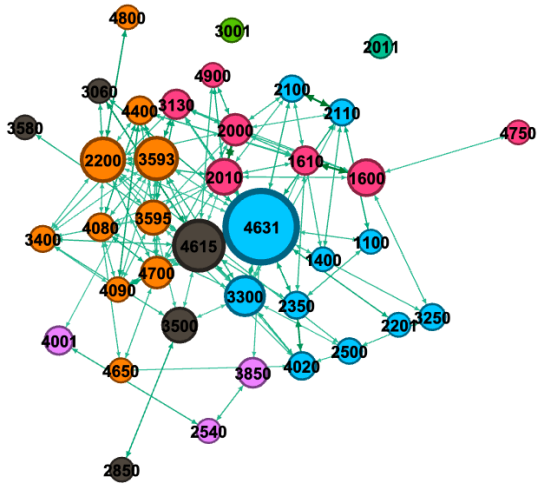


Figure 5: Computer Science network graph.

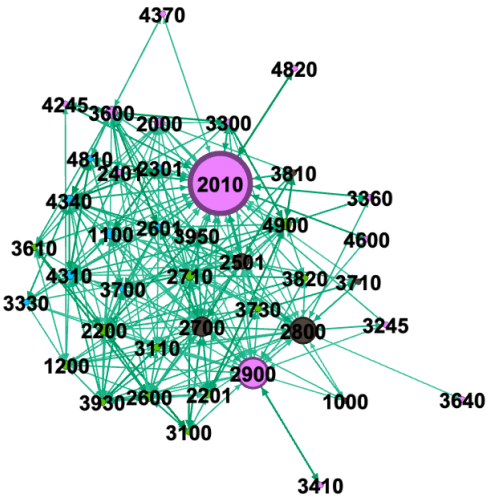


Figure 6: Psychology network graph.

Identifying Hubs in Undergraduate Course Networks Based on Scaled Co-Enrollments

Gary M. Weiss, Nam Nguyen, Karla Dominguez and Daniel D. Leeds
Department of Computer and Information Science
Fordham University, New York, NY
{gaweiss, nnguyen56, kdominguezmelo, dleeds}@fordham.edu

ABSTRACT

This study uses eight years of undergraduate course enrollment data from a major university to form networks of courses based on student co-enrollments. The networks are analyzed to identify "hub" courses often taken with many other courses. Two notions of hubs are considered: one based on raw popularity and another on proportional likelihoods of co-enrollment with other courses. Network metrics are calculated to describe the course networks. Academic departments and high-level academic categories (e.g., humanities), are studied for their influence over course groupings. The identification of hub courses has practical applications, since it can help better predict the impact of changes in course offerings and in course popularity, and in the case of interdisciplinary hub courses, can be used to increase or decrease interest and enrollments in specific academic departments and areas.

Keywords

Graph mining, network analysis, educational data mining.

1. INTRODUCTION

Universities typically offer thousands of different courses across dozens of departments. The interrelationships between courses that are taken together, especially those in different departments, is often not well understood. This paper addresses this deficiency by forming course networks, connecting courses often taken by the same students. Each course is represented as a node in the graph. Several network analyses are pursued. This work also studies "hub" courses, defined as network nodes that are connected to many other nodes, resulting in a high degree count [2]. This study utilizes three popular centrality metrics to identify course hubs and compares the results when using each metric.

Network analyses utilized in this paper have been proven useful to other domains. Analysis of social networks like Facebook identify hubs corresponding to influencers with an outsized impact on other users' purchasing behaviors [3]. Network analysis metrics pursued in the present work have been applied to the World Wide Web, particularly for web searches [5, 8, 9].

Identifying and analyzing hub courses can provide concrete benefits. Courses heavily associated with other courses can be used for better resource planning, particularly when changes are made in the frequency or capacity of such courses. Furthermore, hub courses may be adjusted to drive (or diminish) student interest in an area or academic discipline. For example, there is a current need for more STEM (Science, Technology, Engineering, and Math) professionals. If a hub course is well connected to STEM courses, promoting this course may lead to increased STEM enrollments—even if the hub course is not a STEM course.

The course network analyzed in this study is based on eight years of undergraduate student course enrollment data from Fordham University. An edge connects two courses if the number of students taking both courses is above a threshold. Two types of thresholding mechanisms are considered: (1) a static threshold that is the same for all pairs of courses and (2) a dynamic threshold set to link together only courses taken together *relatively* frequently (i.e., relative to their popularity). We find the dynamic threshold shifts hub courses from humanities to STEM disciplines. Also, tighter course groupings are found within STEM and looser groupings within the humanities and social sciences. An extended version of this paper is available [12].

2. DATASET DESCRIPTION

Our study uses course enrollment data to generate a course-pair dataset, which is then used to form the course networks analyzed in this paper. This course enrollment data contains eight years of undergraduate data from Fordham University, where each record corresponds to one student in one course section. Student grades are also available and used in two of our other studies, one of which analyzes the impact of course sequencing on student grades [4], and the other that forms course networks based on the correlation of grades between courses, and then analyzes the networks [6]. This later study performs a somewhat similar analysis to the one provided in this paper, but with a very different notion of course similarity/linkage.

The course-pair dataset aggregates the course enrollment data to the course level and then extracts information about each course pair. Each course-pair record includes identifying information about two courses and the number of students that took each course and both courses (not necessarily at the same time). The department associated with each course is mapped to one of the six major course categories. The course-pair dataset contains 78,173 records, which are formed from 1,763 distinct courses. The dataset does not contain all possible pairings because pairs with fewer than 20 common students are excluded. The course-pair dataset, and the network metrics provided later, are generated from the course enrollment data using a publicly available Python-based software tool developed by our research group [10].

3. NETWORK ANALYSIS METRICS

The course-pair dataset is used to form course networks by viewing each course as a node and connecting nodes that have a sufficient number of common students. Table 1 provides the network analysis metrics used in this paper. The first three, density, diameter, and average clustering coefficient (ACC) [1], are computed

Gary Weiss, Nam Nguyen, Karla Dominguez and Daniel Leeds "Identifying Hubs in Undergraduate Course Networks Based on Scaled Co-Enrollments". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 809-813. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

using an entire network or subnetwork. Our data shows subnetworks of courses within single departments have a higher density, smaller diameter, and higher average clustering coefficient than the network based on all undergraduate courses, because courses within a discipline are more tightly connected (see Table 2). The last three metrics are defined for each *node* in the network and can be used to help identify hubs. These metrics consist of three centrality measures: degree centrality, eigenvector centrality [11], and betweenness centrality [7]. Each measure can be used to identify a different type of hub course.

Table 1. Summary of network analysis metrics

Metric	Summary Description	Range
Density	Fraction of possible edges present.	0 - 1
Diameter	Maximum distance between any pair of nodes in network.	$\mathbf{Z}^+$
Ave. Clustering Coefficient	Fraction of pairs of neighbor nodes that are connected to each other.	0 - 1
Degree Centrality	Number of edges to node (degree).	$\mathbf{Z}^+$
Eigenvector centrality	Based on centrality of node's neighbors.	≥ 0
Betweenness centrality	Measure all shortest paths passing through node.	≥ 0

4. EDGE INCLUSION METHODOLOGY

To form a course network, each course is represented by a node, and an edge is added between two nodes if the courses, across all sections, have enough common students. Static and dynamic thresholds specify a minimum number of common students.

The static threshold is based on the number of common students between two courses, independent of how many students take each course. The distribution of common students by course pair is provided in Figure 1 in the appendix. Most course-pairs have very few common students, since few students take upper-level courses in disparate disciplines. A threshold of 20 students maintains 11% of all course-pairs with at least one student in common, and this is the static threshold utilized in this study. The static threshold is heavily biased towards popular courses, taken very frequently, even if only a few students in the popular course take specific other courses.

We also define a dynamic threshold relying primarily on the *co-occurrence rate* of courses. The dynamic threshold is determined by multiplying the co-occurrence threshold rate k by the number of students in the larger course within each course-pair. To ensure a minimum number of common students, a static threshold of 20 students is used as the floor for the dynamic threshold. The dynamic threshold, *d-thresh*, associated with two courses, C_1 and C_2, is provided in Equation 1, where C_x.students represents the number of students who have taken class C_x.

$$\text{d-thresh}(C_1, C_2) = \max(20, k \times \max(C_1.\text{students}, C_2.\text{students})) \quad [1]$$

The dynamic threshold is heavily dependent on the co-occurrence rate k, defined as the number of common students divided by the number of students in the larger course. The co-occurrence rate distribution is displayed in Figure 2 of the appendix, which shows that a co-occurrence rate threshold $k = 0.017$ discards 39% of the edges that satisfy the static threshold. This threshold is used because it leads to the most stable centrality measures while excluding the fewest number of edges. Table 5 in the appendix shows how this dynamic threshold impacts an Art History course.

5. RESULTS

This section analyzes course networks using the metrics presented in Table 1 and through the identification of hub courses. Static and dynamic thresholds are considered. Hub results are analyzed within academic departments and broader course categories. This study utilizes six course categories: Arts, Communication and Media Studies, Humanities, Modern Languages, Social Sciences, and STEM. The mapping from academic department to course category is partially provided in Table 2.

5.1 Network Metric Results

Table 2 presents the values of the previously defined network metrics for the course network and subnetworks at the department and category levels. Course categories are denoted in bold, with a subset of two selected associated departments listed below it (see [12] for the full table). The category level value reflects the median values across the member departments. The first row of data provides the values over all courses in the course network. The color of the cells reflects the magnitude of the cell value, with red (green) used for the highest (lowest) values. The colors for the departments and categories are determined independently.

The network covering all courses has a high diameter and low density compared to the subnetworks, since it includes many diverse courses that are loosely connected. Courses associated with a specific department are typically associated with a major; students within the major will take many of these courses. The dynamic threshold decreases the density, average clustering coefficient, and number of edges, while increasing the diameter.

Study of departmental subnetworks shows dynamic thresholding most dramatically decreases edges for Philosophy (52% decrease), English (44% decrease), and Theology (35% decrease), which are fields of study that include many core curriculum courses. This drop is mirrored by ACC. Conversely, the diameter maintains similar values for most departments, regardless of threshold. Overall, dynamic thresholding has a substantial impact on density and ACC of Humanities and Social Science courses, and only minimal impact on other categories, likely reflecting the core curriculum's emphasis on humanities and social science courses.

The STEM courses have much higher density and form much more dense clusters (based on ACC) than humanities courses, for both thresholds. This indicates that humanities students are less likely to take the same group of courses in their discipline. In our university, humanities majors have fewer required courses than STEM majors. Humanities departments have the highest number of nodes (distinct courses taken), closely followed by Social Science, suggesting that those disciplines allow more flexibility in course choices. The Modern Languages category also has a relatively high density and ACC. Language courses, like science courses, typically rely on prerequisite course requirements for proper student preparation.

5.2 Hub Analysis

Hubs play a special role in network structures and play an important role in understanding and utilizing the information in course co-enrollment networks. Table 3 identifies the top-17 hubs using the median of the ranks of the three centrality metrics, "Combined Rank". The top half of the table provides the top-7 hubs when using the static threshold, while the bottom half provides the top-7 for the dynamic threshold. Note that the best combined rank when using the dynamic threshold is 3—no course consistently ranks above third on all the centrality metrics. While only the combined rank for the static (dynamic) threshold is used

Table 2. Summary course network statistics based on category and selected departments

Category/ Department	Nodes	Static Threshold				Dynamic Threshold			
		Edges	Density	Diam.	ACC	Edges	Density	Diam.	ACC
ALL	1763	39968	0.03	4	0.74	24323	0.02	6	0.40
Arts	**41.5**	**239**	**0.32**	**3**	**0.56**	**231**	**0.29**	**2.5**	**0.56**
Dance	54	1236	0.86	3	0.95	1236	0.86	3	0.95
Music	24	87	0.32	3	0.51	73	0.26	2	0.52
Comm and Media Studies	**24**	**25**	**0.20**	**2**	**0.16**	**25**	**0.19**	**2**	**0.16**
Comm and Media Studies	94	862	0.20	3	0.72	828	0.19	4	0.58
New Media & Digital Design	6	8	0.53	2	0.00	8	0.53	2	0.00
Humanities	**81**	**179**	**0.06**	**3**	**0.21**	**104**	**0.04**	**3**	**0.08**
African & African Amer Studies	28	34	0.09	2	0.11	34	0.09	2	0.11
English	167	462	0.03	3	0.59	258	0.02	3	0.12
Modern Languages	**9**	**19**	**0.53**	**2**	**0.49**	**19**	**0.53**	**2**	**0.38**
Greek	4	6	1.00	2	0.00	6	1.00	2	0.00
Spanish	40	118	0.15	3	0.49	98	0.13	2	0.33
STEM	**34**	**295**	**0.47**	**3**	**0.76**	**288**	**0.45**	**3**	**0.75**
Biological Sciences	30	274	0.63	2	0.77	274	0.63	2	0.77
Physics	38	286	0.41	4	0.73	277	0.39	3	0.74
Social Science	**74**	**329**	**0.18**	**2.5**	**0.51**	**285**	**0.16**	**2.5**	**0.44**
Economics	45	325	0.33	2	0.64	270	0.27	2	0.59
Sociology	90	236	0.06	3	0.37	206	0.05	3	0.30

to select the entries in the top (bottom) half of the table, both combined ranks are provided to help compare differences between the thresholding mechanisms. Courses exhibit very different ranks for the two thresholds.

The first few entries for the static threshold in Table 3 vary only slightly depending on which of the three centrality metrics is used. The first four entries cover core curriculum requirements that can only be satisfied by a single course. Most of the remaining top hub courses also satisfy a core requirement, but can be satisfied by several courses. The very few STEM courses listed are introductory and satisfy a core requirement (e.g., *Finite Mathematics*). Thus, we see that hubs identified using the static threshold are based on raw popularity. Most courses identified using the dynamic threshold also satisfy a core requirement, but often many courses can satisfy the requirement. There are no courses that appear in the top-7 lists for both thresholds. For static threshold hubs, most connections to other courses may be incidental, due to so many students taking the popular course.

Table 3. Top-7 static and dynamic course hubs

Courses	Combined Rank		Centrality Rank		
	Static	Dyn.	Deg.	Btw.	Eig.
Static Threshold: Top Hubs					
Philosophical Ethics	1	45	1	1	2
Faith & Critical Reason	2	76	2	2	1
Philos. of Human Nature	3	75	3	3	3
Composition II	4	78	4	5	4
Banned Books	5	49	5	4	5
Finite Mathematics	7	56	6	7	7
Spanish Lang and Lit	7	29	7	6	8
Dynamic Threshold: Top Hubs					
Biopsychology	31	3	3	20	3
Phys. Sci.: Today's World	30	4	4	2	63
Latin American History	44	5	2	6	5
Intro World Art History	22	5	5	26	2
Intro Phys. Anthropol.	41	6	1	9	6
Intro Cultural Anthropol.	18	6	6	33	1
Films of Moral Struggle	55	8	8	7	54

Table 3 allows further comparisons among the centrality metrics. When using the static threshold, course ranks are quite consistent across all the centrality metrics. This ensures that the combined rank is also highly correlated with each of the individual metrics, and that the degree centrality is usually equal to the combined rank. This correlation is weaker when examining the dynamic threshold; degree centrality sometimes differs substantially from the combined dynamic rank; *Calculus II* has degree 7 and combined rank 19. Nonetheless, degree centrality is still generally close to combined rank and is identical in 5 of the first 7 cases.

We focus on degree centrality as our metric for identifying hubs under both thresholds. This is attractive since degree centrality is the simplest and most common metric for identifying hubs. We utilize a degree count threshold of 200 to identify hub courses. This retains all entries in Table 3, which have degree count of at least 245 [12]; the underlying data ensures that a degree count of 200 will retain the top fifty courses associated with each metric).

Table 4 shows the distribution of hub edges between the six course categories using a degree centrality threshold of 200, helping to consider connections across categories. The table displays the percentage of total hub edges from one category (row) to both hub and non-hub courses in another category (column), for each threshold. The percentage of total edges, as well as the actual number of edges, associated with each category (row), are also provided. A color scale is applied to the rows to highlight where the hub connections are directed (red is high percentage and green low percentage). For example, the first row indicates that, using a static threshold, 5% of all Arts hub courses are connected to other Arts courses and 14% are connected to Communication courses. Furthermore, Arts courses have 1,520 edges, comprising 5% of all edges in the course network.

Table 4 shows that for both thresholds, Humanities, STEM, and Social Sciences have the most hub edges, while Arts, Communications, and Modern Language have many fewer. Notably, the static threshold associates more edges with humanities courses than STEM courses (35% to 27%), whereas the dynamic threshold

Table 4. Percent distribution of hub edge linkage by course category (hubs with degree ≥200) with edge info

Category	Static threshold								Dynamic threshold							
	Arts	Comm	Hum	Lang	STEM	SocSci	#Edges	%Edges	Arts	Comm	Hum	Lang	STEM	SocSci	#Edges	%Edges
Arts	5	14	27	7	26	22	1520	5	5	9	21	10	36	20	650	5
Comm	11	24	22	7	20	15	1426	5	8	27	19	9	21	15	954	7
Hum	10	12	31	6	21	20	10892	35	6	11	20	10	33	21	2758	19
Lang	10	16	28	5	18	23	3219	10	9	17	23	5	22	24	1773	12
STEM	5	8	27	7	32	20	8479	27	5	6	24	8	36	21	5543	39
SocSci.	7	10	25	8	25	25	5543	18	3	6	23	10	29	29	2717	19

reverses this trend (19% humanities to 39% STEM). Most core curriculum requirements are associated with humanities and the dynamic threshold has an outsized impact removing courses that are hubs simply due to their popularity.

It is especially notable that more edges link humanities to STEM courses than to other humanities courses. Examining the underlying data, we find that the humanities courses *Introduction to Cultural Anthropology*, *Introduction to Physical Anthropology*, and *Introduction to Art History* all connect to STEM hub courses. Similarly, most connections for courses in the Anthropology and Art History departments go towards the Biological Sciences and Natural Science departments. While *Introduction to Physical Anthropology* is part of the Natural Science major requirement, it also satisfies a science core curriculum requirement for non-Science majors. It is interesting to observe this course's popularity with Science students. The course is a general survey of the biological focus of Anthropology.

Also notable is that Communications and Social Sciences have more links to themselves than to any other category, for both static and dynamic thresholds, even though these categories do not have as many total links as other categories. The Languages category have mostly internal links, and an intermediate number of edges overall for both thresholds.

Social Science hubs in Table 4 have a significant number of connections to STEM courses, commensurate with connections back towards Social Science. Most of the connections to STEM refer to courses in Biological Sciences, particularly from the Psychology course *Foundations of Psychology*. This course is a requirement for the Psychology major but is not part of the core curriculum. This course also has a significant number of connections with the Natural Science department. Overall, the number of connections from non-STEM to STEM courses when using the dynamic threshold is a bit of a surprise. Conversely, STEM hubs made many connections to the Social Science category in Table 4; these connections are largely directed towards the Economics department, which requires a strong mathematical base.

6. CONCLUSIONS

This study analyzed course network graphs using eight years of undergraduate course-grade data from Fordham University. General network statistics and course hub statistics were generated using a publicly available Python-based tool created by our research group [10]. Network structure and hub identity are strongly influenced by the definition of edges between courses, and whether static or dynamic threshold were applied to course co-enrollments. We gain important insights on relations among courses, departments, and categories, and on metrics naturally applied to characterize these relations.

All three common network centrality metrics (degree centrality, betweenness centrality, and eigenvector centrality) identify a similar set of hub courses using static thresholding to define edges. However, the metrics behave much less similarly when dynamic thresholding is used, requiring careful consideration in future analyses. Nonetheless, degree centrality yields a reasonable approximation of the other two metrics for both thresholds, favoring its future use to study course co-enrollment networks.

The static and dynamic thresholds yield very different course networks and hubs. Static thresholds place more emphasis on course popularity, highlighting courses that uniquely satisfy a core requirement. The dynamic threshold reduces, but does not eliminate, popularity bias. Due to the many mandatory humanities core courses, and the variety of core options in STEM, the dynamic threshold substantially shifted apparent hub focus from Humanities to STEM. Future analyses of course relations and discipline relations must continue to carefully weigh the influence of popularity or the mandatory nature of courses. For both thresholds, STEM courses have the highest density and form tightly connected clusters, while humanities courses have the opposite behavior; this is likely due to the more extensive use of prerequisites in STEM disciplines in our university.

Our analysis also identified large numbers of edges between the different course categories. Edge distributions shifted between thresholds, favoring humanities for the static threshold and STEM for the dynamic threshold. Study of courses forming individual edges provided additional insights. The strong connection between humanities and STEM courses was driven by humanities courses like *Introduction to Physical Anthropology*, which has a strong STEM component; the connection between social sciences and STEM was driven by courses like *Foundations of Psychology* which is linked to STEM courses in Biology (Psychology students must take several biology courses).

This study provides a better understanding of course co-enrollment patterns, suggesting directions for valuable practical applications. Strong models of co-enrollment patterns can help with course planning and ensuring enough of course sections are offered. Our course networks reveal valuable details and quantitative relationships among courses. This work is a foundational step in better understanding course co-enrollments.

There are many ways in which this work can be extended and improved. The dynamic threshold could incorporate underlying probabilities of each course being taken, so courses are linked only where their co-occurrence is much more likely than chance. We also can consider additional methods for clustering courses. Future analyses may extend to course ordering information. It may be useful to reduce the influence of popular departments in repeated analysis of category-level network patterns. More fundamentally, our present results may be validated by partitioning the underlying student enrollment records into distinct subsets, to create training and testing data for our network models.

7. REFERENCES

[1] Arif, T. 2015. Mining and Analyzing Academic Social Networks, *International Journal of Computer Applications Technology and Research*, 4, 878-883. 10.7753/IJCATR0412.1001.

[2] Barabási, A. 2016. *Network Science*, Cambridge University Press.

[3] Catanese, S. A., De Meo, P., Ferrara, E., Fiumara, G., and Provetti, A. 2011. Crawling Facebook for social network analysis purposes, *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, 1-8.

[4] Gutenbrunner, T., Leeds, D.D., Ross, S., Riad-Zaky, M., and Weiss, G.M. 2021. Measuring the academic impact of course sequencing using student grade data. In *Proc. of the 14th International Conference on Educational Data Mining.*

[5] Kleinberg, J. M. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46.5 (1999): 604-632.

[6] Leeds, D. D., Zhang, T., and Weiss, G. M. 2021. Mining course groupings using academic performance. In *Proceedings of the 14th International Conference on Educational Data Mining.*

[7] Marsden, P. V. 2005. *Encyclopedia of Social Measurement*.

[8] Page, L., Brin, S., Motwani, R., and Winograd, T. 1999. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab.

[9] Park, H. W., and Thelwall, M. 2003. Hyperlink analyses of the World Wide Web: A review. *Journal of Computer-Mediated Communication*, 8(4).

[10] Riad-Zaky, M., Weiss, G.M., and Leeds, D.D. Course Grade Analytics with Networks (CGAN) [computer software], April 2021.

[11] Ruhnau, B. 2000. Eigenvector-centrality a node-centrality? *Social networks*, 22, 357–365.

[12] G. M. Weiss, N. Nguyen, K. Dominguez, and D. D. Leeds. Identifying hubs in undergraduate course networks based on scaled co-enrollments, arXiv:2104.14500 [cs.SI]

APPENDIX

Figure 1 shows the distribution of common students by course pair (each bin covers a range of common students). The orange curve is a cumulative curve that corresponds to the y-axis values listed to the right (varying between 0% and 80%) and represents the percentage of course-pairs that are *maintained* for each common student threshold value (e.g., a threshold of 20 maintains 11% of all course-pairs with at least one student).

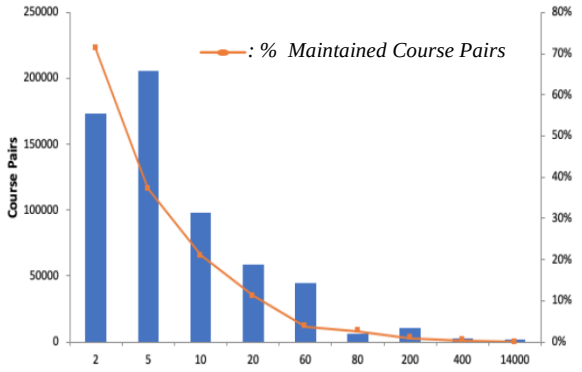


Figure 1. Distribution of common students by course-pair

The dynamic threshold is heavily dependent on the co-occurrence rate *k*. To help set this value appropriately, Figure 2 shows the distribution of course-pairs for each co-occurrence rate, for the course pairs that satisfy the static threshold of 20. The co-occurrence rate is the number of common students divided by the number of students in the course with more students. The co-occurrence rate distribution is heavily skewed to the smaller values, just as the number of common students was skewed to the smaller values in Figure 1. The bar at the far right at x=1.0 is associated with course pairs with the same course in both positions and should be ignored. After some experimentation we decided on a co-occurrence rate threshold $k = 0.017$, which is the value that leads to the most stable centrality measures while excluding the fewest number of edges. The orange curve, which shows the fraction of edges discarded, indicates that this value of *k* discards 39% of the edges that satisfy the static threshold.

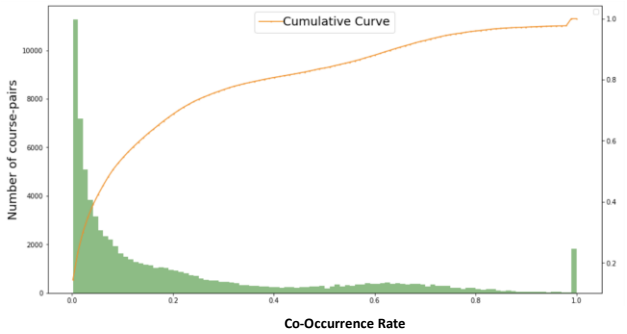


Figure 2. Co-Occurrence Rate Distribution

To illustrate the dynamic threshold, we apply it to the course *Art History Seminar*, which has 123 students. There are 22 courses that share at least 20 students in common with this course, satisfying the static threshold. However, 9 courses have fewer common students than the computed dynamic threshold, and hence are pruned. Half of these 22 courses are displayed in Table 5, and five of these, denoted in **bold**, are pruned since the number of common students is less than the dynamic threshold. As anticipated, the courses affected by the dynamic threshold have a large number of students (third column). In this example, every course that satisfies the static threshold, but is pruned by the dynamic threshold, fulfills a core curriculum requirement.

Table 5. Dynamic threshold for Art History seminar course

Course2	Common Students	Students Course2	Dynamic Threshold
Intro Cultural Anthro.	**23**	**2514**	**43**
Ancient American Art	21	34	20
17th Century Art	22	47	20
20th Century Art	43	130	20
Age of Cathedrals	20	39	20
Aztec Art	22	61	20
Composition II	**58**	**12446**	**211**
Intermediate French II	20	1329	23
Finite Math	**42**	**4976**	**85**
Philosophical Ethics	**58**	**11218**	**191**
Faith & Critical Reason	**56**	**13317**	**226**

Linguistic Features of Discourse within an Algebra Online Discussion Board

Michelle P. Banawan
Arizona State University
Tempe, AZ
mbanawan@asu.edu

Renu Balyan
SUNY, Old Westbury
New York
balyanr@oldwestbury.edu

Jinnie Shin
University of Florida
Gainesville, FL
Jinnie.Shin@coe.ufl.edu

Walter L. Leite
University of Florida
Gainesville, FL
Walter.Leite@coe.ufl.edu

Danielle S. McNamara
Arizona State University
Tempe, AZ
dsmcnama@asu.edu

ABSTRACT

This study leverages natural language processing to assess dimensions of language and discourse in students' discussion board posts and comments within an online learning platform, Math Nation. This study focusses on 1,035 students whose aggregated posts included more than 100 words. Students' wall post discourse was assessed using two linguistic tools, Coh-Metrix and SEANCE, which report linguistic indices related to language sophistication, cohesion, and sentiment. A linear model including prior math scores (i.e., Mathematics Florida Standards Assessments), grade level, semantic overlap (i.e., LSA givenness), incidence of pronouns, and noun hypernymy accounted for 64.48% of the variance for the Algebra I end of course scores (RMSE=13.73). Students with stronger course outcomes used more sophisticated language, across a wider range of topics, and with less personalized language. Overall, this study confirms the contributions of language and communication skills over and above prior math abilities to performance in mathematics courses such as Algebra.

Keywords

Student performance, performance prediction, discussion posts, linguistic features

1. INTRODUCTION

Discussion boards have emerged to be among the most beneficial features of online learning platforms. Some of the positive outcomes obtained include greater student involvement and improved academic performance [1-5]. Discussion boards have been implemented to achieve a number of educational goals, namely, to supplement course resources, evoke creativity and motivation, facilitate interaction between teachers and learners, and for class management or administrative purposes [6-9]. Student engagement and collaboration within discussion boards are critical towards their success. Indeed, students' language used within these discussion boards has been linked to positive learning outcomes [10-12]. This creates a pressing need to further understand the language used by students when collaborating with each other or engaging with their teachers within informal online academic settings.

Michelle Banawan, Renu Balyan, Jinnie Shin, Walter Leite and Danielle McNamara "Linguistic Features of Discourse within an Algebra Online Discussion Board". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 814-819. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

1.1 Language and Math Success

While empirical evidence shows mixed results in the correlations between language proficiency and academic success (i.e., some found significant correlations and some none), proponents have articulated that language proficiency, and more importantly - communicative competence significantly influence success in math [13]. The dimensions of language that have been found to specifically influence math achievement include linguistic complexity, language control, and vocabulary usage [14].

A number of studies have demonstrated links between language and performance in math [10, 11, 15]. There are strong links between language skills and the ability to engage with math concepts and problems. For instance, success in math is partially based on the development of language that affords children the ability to participate in math instruction in the classroom as well as "engage quantitatively with the world outside the classroom" [16]. Similarly, strong math skills are presumed to interact with language ability to understand numbers and symbols [17]. Linguistic skills may be one of the key factors that relate to math ability. For instance, Cummins identified language difficulties in second language speakers as a key obstacle in solving math problems [18]. Articulating and representing cognitive processes in math domains is especially challenging for students with lower literacy skills. Successfully solving verbal analogies and mathematical word problems, in particular, demand certain levels of linguistic fluency and reading comprehension skills, which can be barriers to success.

More specific to discourse in online discussion boards, substantial work has been done to characterize the language used within online discussion forums [19-24]. This research indicates that linguistic features distinguish subject matter experts from nonexperts and are predictive of student learning outcomes. In social questioning and answering sites (e.g., Quora), linguistic features such as word usage, average number of words, subjectivity of words, and word complexity have been found to be markers of expertise [19]. Discourse analyses conducted on online discussion boards show that linguistic characteristics are predictive of student learning performance [20]. To name a few, the complexity of syntactic structures, cohesion, emotion words, modal verbs, and words that provide additional information or make claims when elaborating are significant predictors of students' performance [20-24].

1.2 Current Study

This study examines students' discussion wall posts for an entire academic year within an online Algebra tutoring platform, Math Nation, developed by the University of Florida Lastinger Center for Learning [25-29]. Math Nation is an interactive and comprehensive math teaching and learning platform that provides video tutorials and online resources aligned to the Mathematics Florida Standards (MAFS). Most relevant to the current study, Math Nation also features an online discussion forum called Algebra Wall where students can collaborate with other students, teachers, and study experts. Wall posts (see Figure 1 for a sample discussion thread) from 3,277 students, Math Nation study experts, and Algebra teachers were collected for the period August 1, 2018 to July 31, 2019, including comments within more than 14,000 threads.

Our objective in this study was to further examine the extent to which the linguistic features of these posts were predictive of End of-Course (EOC) Algebra performance, over and above their scores Math scores from the previous year. Providing information concerning students' potential EOC performance is important because it has the potential to augment stealth assessment of students' abilities such that the instructor or the tutoring system can intervene and provide scaffolding when necessary.

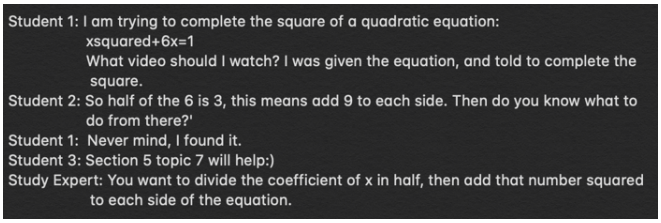


Figure 1. Sample Discussion Thread

2. METHODS

2.1 Participants

The participants included 3,277 Algebra students from the different Florida school districts in grade levels 7, 8. and 9 who participated in the Math Nation discussion board for the academic year August 1, 2018 to July 31, 2019. The majority of these students were white (n = 2,464, 75%). This study focusses on 1,035 students in this larger sample whose aggregated posts that included more than 100 words because NLP indices are not reliable with small language samples, and many of our indices (e.g., lexical diversity) require a minimum of 100 words [32]. Those who included more words in their posts had significantly higher FSA scores, $t(3275) = 5.79$, $p<.001$ ($M_{\leq 100\ words}$,= 354.08, SD = 16.83; $M_{>100\ words}$,= 357.75, SD = 16.89); and EoC scores, $t(3275) = 8.12$, $p<.001$ ($M_{\leq 100\ words}$,= 522.66, SD = 22.99; $M_{>100\ words}$,= 529.67, SD = 22.93). As such, number of words in posts are strong indicators of future and current math performance; the purpose of this study is to examine language beyond number of words.

*2.1.1 **Prior Math and Algebra I EoC Scores***

Students' mathematics performance was measured using Algebra I End-of-Course (EoC) assessment (M=524.88; SD=23.20; Range = 425-575), which is a high-stakes exam required by the Florida Department of Education [30] for high school graduation. Mathematics Florida Standards Assessments (FSA) scores (M=355.24; SD=16.93; Range = 269-393), from the previous year were included as proxy baseline scores indicative of Math preparedness [31]. The FSA math scores are often used as a baseline measure of Algebra skills or preparedness because they are strongly related to the students' Algebra I EoC scores. Indeed, the relation between these two tests was strong in the current study ($r=0.76$, $p<0.01$). Controlling for gender, grade level and district did not result in significant variations in the correlation between the Math FSA score and the Algebra I score (i.e., r = .73 - .76). Table 1 shows the mean scores for both the FSA Math and Algebra I exams as a function of grade level and gender.

Table 1. Algebra performance

	Math (FSA) Score from Previous Year (Mean / SD)	**Algebra I Score (Mean / SD)**
Grade 7 (n =440)	362.89 (15.74)	539.31 (19.02)
Grade 8 (n = 520)	355.39 (16.04)	525.58 (20.75)
Grade 9 (n = 75)	343.92 (17.95)	501.49 (26.49)
Male (n = 429)	359.92 (16.79)	531.55 (23.25)
Female (n = 606)	356.21 (16.80)	528.33 (22.62)

2.2 Natural Language Processing Tools

We assessed students' Math Nation Wall discourse using two linguistic tools, namely Coh-Metrix [32] and SEANCE [33], which report linguistic indices related to language sophistication, cohesion, and sentiment. Use of these two tools was motivated by prior work relating academic performance in mathematics to these features of language in online forums and discussion boards [20-24].

2.2.1 Coh-Metrix

Coh-Metrix provides multiple levels of linguistic analysis that include indices at word level and sentence level, indices related to connections between the sentences, and discourse relationships between the texts and their mental representations. Coh-Metrix has been used to analyze different forms of text in the English language that are written to communicate messages to readers, including those within tutoring sessions, chat rooms, email exchanges and other forms of informal conversation [34, 35]. In the current study the Coh-Metrix indices that estimate psycholinguistic measures, word information, syntactic patterns, syntactic complexity, situation model, lexical diversity and other descriptive indices were used to specifically investigate the linguistic profiles of discourse in the Math Nation Wall posts.

2.2.2 SEANCE

The Sentiment Analysis and Cognition Engine (SEANCE) calculates sentiment indices for a text using pre-developed word vectors that measure sentiment and pre-existing sentiment, social positioning and cognition dictionaries. One particular advantage of SEANCE is that accounts for the presence of negations in the texts (e.g., *not sad*, would not be assessed as negative). Yoo and Kim found positive emotions reported by SEANCE to be strong predictors of success [35]. SEANCE has also been previously used to model math identity and math success [12]. In another study, Crossley et al. demonstrated using SEANCE that math performance was related to the use of fewer words related to respect [11]. Similarly, we used SEANCE in the current study to assess the extent to which sentiment expressed within the discussion posts was related to math performance.

2.3 Data Preprocessing and Feature Selection

The dataset was checked for multicollinearity as it reduces the precision of the estimate coefficients and makes it difficult to assess the relative importance of the independent variables in explaining the variation caused by the dependent variable. Highly correlated features (r >= 0.90) were removed from the analysis. In case two or more attributes were found to be highly correlated, the attributes with the greater number of pairwise correlations were removed. The dataset was further filtered such that features with more than 20% values were missing and those with zero and nearly zero variance were also removed.

The analysis initially included 124 variables (92 Coh-Metrix features, 20 SEANCE Component Scores, and 12 variables related to student factors). After preprocessing and feature selection, 12 linguistic indices, and 4 student variables were included in subsequent analyses. The 12 features represent cohesion and sentiment measures, whereas the 4 non-linguistic indices represent demographics and performance data.

3. RESULTS

The purpose of this study was to assess the degree to which linguistic features of students' language within the Math Nation Wall posts predicted EOC performance compared to other more traditional measures such as demographics and prior math performance. To this end, three linear models were examined using different combinations of candidate predictors of math performance (i.e., non-linguistic features, linguistic features, and the combination of both the non-linguistic and the linguistic features). The necessary assumptions for testing the regression models were met by examining model residuals for all three models. Figure 2 provides the sample diagnostic plots for the residual analysis of the full model. The residuals versus fitted graph reveal no pattern, show a constant variation, and depict linearity. The normal Q-Q plot also shows normal distribution of the residuals. The remaining 2 plots do not depict any non-linear behavior nor any influence of homoscedasticity. These models were also validated using 10-fold cross-validation which rendered the best fit models in terms of RMSE performance.

3.1 Non-linguistic Predictive Model

The non-linguistic features included in this regression model were the students' gender, grade level, and FSA math scores of the previous year (see Table 2). Using only the FSA math scores as a candidate predictor the resulting model accounted for 58.64% of the variance. Using the FSA math score, gender, and grade level, grade level also emerged as a significant predictor but gender did not. The model with the FSA and grade level as predictors accounted for 63.12% of the variance of the EoC scores. No significant interactions between grade and gender emerged.

These results suggest that the Math FSA score depicting prior performance in mathematics and the grade level significantly contributes to Algebra EoC performance, providing adequate proxies for students' baseline performance prior to the course.

3.2 Linguistic Predictive Model

The primary purpose of this study is to examine the degree to which features of the language used by students in the wall posts are predictive of students' Algebra EoC scores over and above baseline proxies provided by FSA performance and demographic variables.

We conducted a multiple linear regression analysis predicting Algebra I scores using the 12 linguistic indices discussed in Section 2.3.

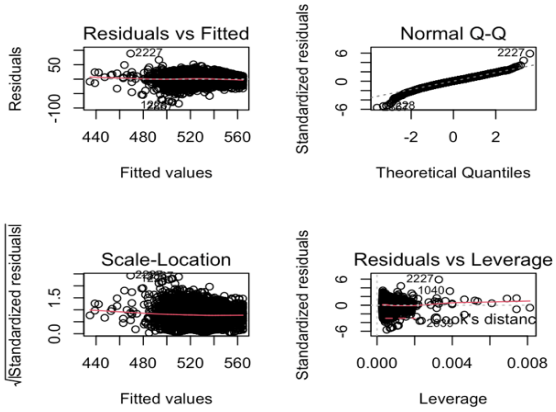


Figure 2 - Diagnostic Plots for linear regressions

Table 2. Linear model including Non-linguistic Features

	Estimate	S.E.	*t*
Using Math FSA only as candidate predictor			
Math FSA Score	1.040	0.027	38.30
Intercept	157.778	9.721	16.23
Using Math FSA, grade and gender as candidate predictors			
Math FSA Score	0.949	0.027	35.052
Grade	-8.395	0.745	-11.262
Gender *	1.000	0.884	1.131
Intercept	253.855	12.691	20.003

Notes: Gender is not significant (p = 0.258); All other p< 0.001. Random-effects were estimated with school district (level 1) in a nested mixed-effects model, resulting in a negligible amount of variance account by the school district (3.67%). Hence, the final models were constructed without district.

Table 3. Linear model including Linguistic Features

	Estimate	S.E.	*t*
Semantic overlap (givenness) of each sentence	-75.698	15.150	-4.997
Incidence of Pronouns	-0.159	0.024	-6.732
Hypernymy for nouns	5.617	0.976	5.758
Intercept	527.486	7.343	71.838

Note: p-value at < 0.001

After the 10-fold cross validation, the best-tuned model accounted for 10.64% of the EOC variance with an RMSE = 21.73. Table 3 reports the coefficients of the significant linguistic. Students whose posts include higher noun specificity (hypernymy) also tended to have higher Algebra I EoC performance. Moreover, lower degrees of sentence givenness and pronoun incidence also emerged as indicators of Algebra I EoC performance. These results imply that the posts by better performing students were structured in such a way that they used more specific terms for concepts or topics (higher noun hypernymy), less personal (lower pronoun incidence), and included queries or responses with greater amount of elaboration on topics that varied across posts (lower sentence givenness/newness).

3.3 Combined Model

The combined model included the significantly predictive features from both the non-linguistic and linguistic models (i.e., FSA score, grade level, LSA givenness, incidence of pronouns, and hypernymy of nouns). This model accounted for 64.48% of the variance for the Algebra I EoC scores with an RMSE of 13.73. The results are summarized in Table 4.

The findings revealed that the full model with the combined linguistic and non-linguistic features performed only slightly better than the baseline model in predicting Algebra I scores (i.e., 63.1% vs. 64.5% of the variance). An ANOVA was conducted to compare the fitness of both regression models, comparing the non-linguistic model to the model with the linguistic predictors. The results indicated that the more complex model with the additional linguistic predictors better captured the variance of the Algebra I EoC scores than the baseline model, $F = 20.879$, $p < 0.001$. We also used Akaike information criterion (AIC) model selection to select the best fit model between the non-linguistic model and the model with the linguistic predictors. The model with the linguistic predictors emerged as the best-fit model carrying 100% of the model weight (AICc weight = 1) and having lower AICc (AICc full model = 8,357.60; AICc baseline model = 8394.72) in predicting Algebra I EOC performance.

These results replicate prior studies [10,15] suggesting students' language fluency and use within Math discussion boards provide valuable information regarding students' potential performance at the end of the year. Importantly, these features can be captured dynamically as the course progresses, and in the absence of other information, such as prior course scores and demographics.

Table 4. Linear model for Combined Features

	Estimate	S.E.	t
Math FSA score	0.902	0.027	32.889
Grade	-8.432	0.731	-11.533
Hypernymy for nouns	2.919	0.617	4.730
Semantic overlap (givenness) of each sentence	-27.529	9.594	-2.869
Incidence of Pronouns	-0.044	0.015	-2.929
Intercept	264.302	13.502	19.575

Note: all $p < 0.001$

4. CONCLUSION

In summary, the results reported in this study confirm prior studies that have suggested that the students' math course scores, and in this case Algebra 1 EoC scores, can be significantly predicted by language, in particular hypernymy, pronoun incidence, and lower semantic overlap between sentences. Students with stronger course outcomes used more sophisticated language, across a wider range of topics, and with less personalized language.

Students' math scores from the previous year served as a proxy for baseline math performance, or prior math skills. As expected, prior math skills provided the strongest predictors of the EoC Algebra I scores. Students' grade level also emerged as a significant (negative) predictor of the Algebra I EoC performance. The students self-select as to when they would take the Algebra I course. As such, higher ability students tend to take Algebra I in middle school whereas lower ability students tend to take the exam later in high school, and thus grade was negatively related to scores.

Hypernymy (specificity) of nouns, an indicator of language fluency, contributed to the prediction of EoC performance such that a higher degree of hypernymy or specificity the words used in the discourse was related to higher EoC scores. Further, the discourse of higher performing students can be characterized as less personal as depicted by lower pronoun incidence. In addition, higher performing students' posts had lower overlap between posts, and more new information as depicted by the lower givenness/new LSA index.

We assume that students' engagement in online discussions reveals some aspects of their mental representations or understanding of the academic content. In turn, the linguistic features of their language can serve as proxies for underlying literacy and math skills. The linguistic features that pertain to language fluency suggest that students' posts were reflective of their ability to communicate more effectively and use terms more specific to the academic content.

Notably, linguistic features depicting sentiment did not emerge as significant predictors of Algebra I EoC performance. This could be attributed to the academic nature of the discussion such that students' discourse tends to be more domain-related and less personal in nature. Yet, there is a strong tendency in the NLP literature to focus on sentiment in language. This study indicates that when other features related to language sophistication are considered, sentiment may not emerge as a significant predictor of performance.

There are multiple implications from this work. The first is relatively obvious: literacy and language skills contribute to students' math performance. Language skills aide in student' comprehension of math and their ability to communicate regarding math. In turn, they are more likely to succeed. As such, providing literacy instruction is important: to enhance students' performance in language courses (ELA), but also for performance in content courses (science, history) and mathematics courses. Second, these results suggest that it behooves educators to consider literacy and communication skills, and provide instructions as concretely and coherently as possible [36-38].

Third, within the context of online platforms, these results further confirm the potential of leveraging linguistic and semantic features of students' posts as indicators of potential course performance. It might be assumed that language is not an important indicator of math; and yet, multiple studies have demonstrated that the linguistic features are powerful proxies for students' underlying skills and knowledge across a variety of contexts. Our future studies will consider other features of language (e.g., rhetorical features, lexical features) as well as examining students' language use across various times during the course. Whereas this study solely examined aggregated posts at the end of the course, our future work will examine the number of posts necessary to significantly predict performance. Dynamic, online predictions are necessary in order to intervene, provide appropriate scaffolding to the students, and usable information for the mathematics instructors. Linguistic dimensions of the production of online discourse and their association with academic performance is a promising field of research. As such, linguistic profiles of discourse have strong potential to inform instructional and pedagogical design of collaborative learning environments such as Math Nation.

ACKNOWLEDGMENTS

The research reported here was supported by the Institute of Education Sciences (IES Grant: R305C160004). Opinions, conclusions, or recommendations do not necessarily reflect the views of the IES. We also acknowledge and thank the developers of Math Nation as well as the many team members on the VLL project.

5. REFERENCES

[1] Ringler, I., Schubert, C., Deem, J., Flores, J., Friestad-Tate, J., & Lockwood, R. (2015). Improving the asynchronous online learning environment using discussion boards.

[2] Delaney, D., Kummer, T. F., & Singh, K. (2019). Evaluating the impact of online discussion boards on student engagement with group work. British Journal of Educational Technology, 50(2), 902-920.

[3] Cole, J. E., & Kritzer, J. B. (2009). Strategies for success: Teaching an online course. *Rural Special Education Quarterly*, *28*(4), 36-40.

[4] Nieuwoudt, J. E. (2020). Investigating synchronous and asynchronous class attendance as predictors of academic success in online education. *Australasian Journal of Educational Technology*, *36*(3), 15-25.

[5] Kauffman, H. (2015). A review of predictive factors of student success in and satisfaction with online learning. *Research in Learning Technology*, *23*.

[6] Covelli, B. J. (2017). Online discussion boards: The practice of building community for adult learners. *The Journal of Continuing Higher Education*, *65*(2), 139-145.

[7] Wright, S., & Street, J. (2007). Democracy, deliberation and design: the case of online discussion forums. *New media & society*, *9*(5), 849-869.

[8] Brush, A. J., Bargeron, D., Grudin, J., Borning, A., & Gupta, A. (2002). Supporting interaction outside of class: Anchored discussions vs. discussion boards.

[9] Farmer, J. (2004, December). Communication dynamics: Discussion boards, weblogs and the development of communities of inquiry in online learning environments. In *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference* (pp. 274-283).

[10] Crossley, S., Barnes, T., Lynch, C., & McNamara, D. S. (2017). Linking Language to Math Success in an On-Line Course. International Educational Data Mining Society.

[11] Crossley, S., Liu, R., & McNamara, D. (2017). Predicting math performance using natural language processing tools. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference* (pp. 339-347).

[12] Crossley, S., Ocumpaugh, J., Labrum, M., Bradfield, F., Dascalu, M., & Baker, R. S. (2018). Modeling Math Identity and Math Success through Sentiment Analysis and Linguistic Features. *International Educational Data Mining Society*.

[13] Graham, J. G. (1987). English language proficiency and the prediction of academic success. *TESOL quarterly*, *21*(3), 505-521.

[14] Grant, R., Cook, H. G., & Phakiti, A. (2011). Relationships between language proficiency and mathematics achievement. *Madison, WI: WIDA Consortium*.

[15] Crossley, S. A., Karumbaiah, S., Ocumpaugh, J., Labrum, M. J., & Baker, R. S. (2020). Predicting math identity through language and click-stream patterns in a blended learning mathematics program for elementary students. *Journal of Learning Analytics*, *7*(1), 19-37.

[16] Vukovic, R. K., & Lesaux, N. K. (2013). The language of mathematics: Investigating the ways language counts for children's mathematical development. *Journal of Experimental Child Psychology*, *115*(2), 227-244.

[17] Adams, T. L. (2003). Reading mathematics: More than words can say. *The Reading Teacher*, *56*(8), 786-795.

[18] Cummins, J. (1979). Linguistic interdependence and the educational development of bilingual children. *Review of educational research*, *49*(2), 222-251.

[19] Patil, S., & Lee, K. (2016). Detecting experts on Quora: by their activity, quality of answers, linguistic characteristics and temporal behaviors. *Social network analysis and mining*, *6*(1), 5.

[20] Yoo, J., & Kim, J. (2014). Can online discussion participation predict group project performance? Investigating the roles of linguistic features and participation patterns. International Journal of Artificial Intelligence in Education, 24(1), 8-32.

[21] Vercellone-Smith, P., Jablokow, K., & Friedel, C. (2012). Characterizing communication networks in a web-based classroom: Cognitive styles and linguistic behavior of self organizing groups in online discussions. *Computers & Education*, *59*(2), 222-235.

[22] Guiller, J., & Durndell, A. (2007). Students' linguistic behaviour in online discussion groups: Does gender matter?. *Computers in Human Behavior*, *23*(5), 2240-2255.

[23] Montero, B., Watts, F., & García-Carbonell, A. (2007). Discussion forum interactions: Text and context. *System*, *35*(4), 566-582.

[24] Zhu, M., Herring, S. C., & Bonk, C. J. (2019). Exploring presence in online learning through three forms of computer-mediated discourse analysis. Distance Education, 40(2), 205-225.

[25] Shin, J., Balyan, R., Banawan, M., Leite, W., & McNamara, D. (Accepted). Pedagogical Communication Language in Video Lectures: Empirical Findings from Algebra Nation. *International Society of Learning Sciences* (ISLS).

[26] Shin, J., Balyan, R., Banawan, M., Walter, L. & McNamara, D.S. (Accepted). Discovering Pedagogical Communication Strategies with Algebra Tutoring videos with a Theory-based NLP Approach. To be presented at American Educational Research Association (AERA: April 2021).

[27] Leite, W. L., Cetin-Berber, D. D., Huggins-Manley, A. C., Collier, Z. K., & Beal, C. R. (2019). The relationship between Algebra Nation usage and high-stakes test performance for struggling students. Journal of Computer Assisted Learning, 35(5), 569-581.

[28] Niaki, S. A., George, C. P., Michailidis, G., & Beal, C. R. (2019, March). Investigating the Usage Patterns of Algebra Nation Tutoring Platform. In Proceedings of the 9th International Conference on Learning Analytics & Knowledge (pp. 481-490).

[29] Algebra Nation. (n.d.). Retrieved January 22, 2021, from https://lastinger.center.ufl.edu/mathematics/algebra-nation/

[30] Florida Department of Education. (2018). Statewide assessment program information guide 2018-2019. Florida Department of Education.

[31] Nelson, J., Perfetti, C., Liben, D., & Liben, M. (2012). Measures of text difficulty: Testing their predictive value for grade levels and student performance. *Council of Chief State School Officers, Washington, DC*.

[32] McNamara, D. S., Graesser, A. C., McCarthy, P. M., & Cai, Z. (2014). Automated evaluation of text and discourse with Coh-Metrix. Cambridge University Press.

[33] Crossley, S. A., Kyle, K., & McNamara, D. S. (2017). Sentiment analysis and social cognition engine (SEANCE): An automatic tool for sentiment, social cognition, and social order analysis. Behavior Research Methods 49(3), pp. 803-821. doi:10.3758/s13428-016-0743-z.

[34] Cade, W., Dowell, N., Graesser, A., Tausczik, Y., & Pennebaker, J. (2014). Modeling student socioaffective responses to group interactions in a collaborative online chat environment. In Educational Data Mining 2014.

[35] Yoo, J., & Kim, J. (2014). Can online discussion participation predict group project performance? Investigating the roles of linguistic features and participation patterns. International Journal of Artificial Intelligence in Education, 24(1), 8-32.

[36] MacGregor, M., & Price, E. (1999). An exploration of aspects of language proficiency and algebra learning. *Journal for Research in Mathematics Education*, *30*(4), 449-467.

[37] Wang, J., & Goldschmidt, P. (1999). Opportunity to learn, language proficiency, and immigrant status effects on mathematics achievement. The Journal of Educational Research, 93(2), 101-111.

[38] Lager, C. A. (2006). Types of mathematics-language reading interactions that unnecessarily hinder algebra learning and assessment. *Reading Psychology*, *27*(2-3), 165-204.

Feedback and Self-Regulated Learning in Science Reading

Effat Farhana[1], Andrew Potter[2], Teomara Rutherford[2], Collin F. Lynch[1]
[1]North Carolina State University, [2]University of Delaware
efarhan@ncsu.edu, ahpotter@udel.edu, teomara@udel.edu, cflynch@ncsu.edu

ABSTRACT

How do students respond to feedback in a reading platform? In this study we examined students' (n = 670) reading and SRL behaviors after receiving feedback from their teachers. First, we examined the extent in which students revised their responses after receiving feedback. Second, we examined the association of reading and SRL behaviors with student scores after feedback. Third, we examined relationships between the type of feedback received (i.e. teacher comments) and subsequent student behaviors. We found that students who revised their answers more had greater score improvements. Teacher feedback in writing conventions was shown to produce fewer reading and SRL behaviors when compared to other types of feedback. The number of reading events was correlated with improved scores, although the effect size was small. These findings suggest that teacher feedback can help students employ reading and SRL behaviors and improve their reading comprehension under the right conditions. We discuss recommendations and possible design implications for online reading platforms.

Keywords

Self-Regulated Learning, Feedback, Sequence mining, Reading comprehension, Natural Language Processing

1. INTRODUCTION

Feedback can improve students' performance [38] and Self-Regulated Learning (SRL) behaviors [10]. However, students must understand feedback in order to apply it [48]. Feedback gaps occur when students receive but do not act upon feedback [24], and may be caused by lack of clarity [9], students' misunderstanding of feedback application [55], and the feedback paradox [61] (i.e., students do not address feedback despite understanding its importance). Researchers have recently emphasized the actionability of feedback as one factor to change students' actions and behaviors [12]. This concept remains largely underexplored [34].

To address the feedback actionability gap, researchers have analyzed how students act upon receiving feedback by examining students' perceptions [37, 50] and analyzing student behavior, including timely response to feedback [34], the effect of different types of feedback on the same question [27], and students' learning strategies usage [43].

Effat Farhana, Andrew Potter, Teomara Rutherford and Collin F. Lynch "Feedback and Self-Regulated Learning in Science Reading". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 820-826. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

We examine students' feedback response behavior in science reading. Science reading skills are of critical importance, but challenging for students to master [63]. Science reading can be enhanced through the application of SRL skill [15, 47]. To investigate SRL and science reading, we conducted our analysis on middle school science readings and questions from an online learning platform, Actively Learn (AL). We answered three research questions:

RQ1. How do students' scores vary after receiving feedback?
RQ1.1. To what extent do students change their answers after receiving feedback?
RQ2. How does students' reading and SRL usage vary upon receiving feedback?
RQ3. Is feedback type associated with subsequent reading and SRL behaviors?

2. RELATED WORK

2.1 SRL and Reading

SRL refers to four regulatory processes during learning: goal setting, self-monitoring, self-evaluating, and applying strategies [65]. Self-regulated learners use self-monitoring skills to monitor their tasks [69] and can judge their learning outcomes in light of their goals [68]. Self-regulation is associated with academic performance [49, 66]. SRL researchers have proposed theories and models (e.g., Pintrich's SRL framework [49], Zimmerman's Cyclic model [66]) to explain learners' SRL behaviors. In this study, we adopt Winne and Hadwin's model [56, 58] to measure students SRLs from students' log trace data within AL, as it has proven a useful framework for similar research [4].

SRL-based reading interventions have been effective in improving middle school reading in experimental studies [54]. Computer-based learning environments (CBLEs) can integrate SRL instruction via features to help students foster SRL skills. Examples of CBLEs that are rooted in models of SRL and have been shown to support reading comprehension and SRL behaviors for reading include iSTART [44-45], nSTUDY [4], and ReaderBench [18-19]. In this study, we examine the web-based platform Actively Learn (AL), which uses platform features that promote SRL (Section 3.1).

2.2 Sequence Mining

Sequence mining techniques can identify students' learning behaviors [2, 29]. For example, *n*-gram sequencing techniques have been applied to a game-based learning platform to identify students' problem-solving behavior [2] and to study associations between students' academic performance and transition behavior among multiple platforms [29].

In this study, we are focused on what SRL activities students engage in on the AL platform *after* they receive feedback on a prior submission. In this analysis we applied an approach used by Sheshadri et al. [52] to examine sequence behaviors across platforms. In this approach we aggregated distinct SRL and question submission actions within AL and then examined the frequency and sequence of the activities prior to a resubmission.

2.3 Feedback

Providing feedback and opportunities for students to respond to feedback is one way teachers can assist their students in reading to learn tasks in STEM domains [42]. However, feedback quality can influence students' responses [53]. Hattie and Timperley [31] characterized feedback at four levels: the task (i.e., how well the student accomplished a task), processing (i.e., the processes required to complete the task), self-regulation (i.e., how students choose and implement self-regulatory strategies to accomplish a task); and the self-level (i.e., personal evaluations). Feedback effectiveness is also moderated by the amount of information provided; different types of feedback should be considered as separate constructs [60]. Prior studies have also shown that timely engagement with personalized feedback was associated with academic success for undergraduate students [34] and may also prompt more engaged learning activities when compared to general feedback [43].

Corrective and self-regulatory feedback given to students after answering comprehension questions in response to texts in a digital environment can enhance SRL behaviors and performance [39, 40]. In an experimental study, students who received self-regulatory feedback made more text searches and included more textual info in their responses when compared to students who received less informative or no feedback [39]. A follow-up study replicated these findings and also demonstrated that requiring students to select relevant text information before re-submitting answers led to improved SRL behaviors [40]. Taken together, these studies suggest that corrective and self-regulatory feedback can improve SRL behaviors and reading performance when students are tasked with re-submitting answers to comprehension questions with digital texts.

However, it can be challenging for teachers to provide timely and informative feedback at scale [31, 53]. A prior study on feedback comments of science assignments [8] indicated that feedback that did not provide a correct answer was only helpful if students knew where to find the correct answer; more informative feedback was required when students lacked background knowledge Prior research also suggests that timely engagement with feedback, particularly personalized feedback, was associated with academic success for undergraduate students [34]. Written comments can provide an effective means for providing feedback on science content [8] and in digital reading comprehension tasks [39, 40]. In this study we examined how teacher feedback comments within the context of a digital science reading comprehension related to students' SRL behaviors.

3. Actively Learn (AL) Platform

AL is an online K-12 reading platform for multiple disciplinary subjects. AL catalogs curriculum-integrated readings that teachers can assign as in-class or homework assignments. Teachers can also add their own content as assignments. AL assignments contain text-embedded questions that can be multiple choice (MCQ) and short-answer (SA), including open ended questions and fill-in-the blanks. Teachers can give feedback on students' answers to questions by scoring questions on a scale from 0-4 and writing comments.

We adopted Winne and Hadwin's SRL model in our study. Winne and Hadwin's model has four phases: task defining (**Phase 1**), goal setting (**Phase 2**), enacting tactics and strategies (**Phase 3**), and metacognitively adopting strategies (**Phase 4**). We primarily focus on students' usage of SRL tactics/strategies within AL (**Phase 3)** and adapting reading and SRL (**Phase 4)** upon receiving feedback. Our study is grounded in the Winne and Hadwin model, as its focus on the events underlying SRL [57] fits the retrospective analysis of student interaction data within our study. Furthermore, we focus on three types of SRL events that are consistent with prior literature situated in Winne and Hadwin's model: annotating [3, 41], highlighting [59], and vocabulary lookups [5].

3.1 Dataset Preparation

The present study was conducted with middle school physical science data collected from AL in 2018. The initial dataset contained 17,886 student records from 1,033 classes. First, after data cleaning, we included classes containing 10-60 students (n = 14,925 students). Second, we identified student submissions on which they received feedback. This reduced dataset included 1,819 unique students, 3,867 questions, and 5,373 submissions. Third, we applied the following filtering criteria: 1) a student submitted a question multiple times, 2) received at least one instance of feedback, and 3) re-submitted after receiving feedback. The trimmed dataset, which included student empty submissions, contained 670 unique students in 113 classes, 58 teachers, 156 assignments, 1,072 questions, and 2,502 submissions. All questions in our dataset are SA questions.

4. METHODOLOGY

We describe our methodology for each RQ in this section.

4.1 RQ1 Methodology

To answer RQ1 we measured students' score differences by calculating the difference between the first and last submission scores when students made multiple attempts after receiving feedback. We observed three categories of submissions: score increased in the last submission, score decreased in the last submission, and score was unchanged in the last submission.

To assess whether students were addressing teachers' feedback, we calculated similarity between subsequent answer submissions of a question. We hypothesized that changes in submitted answers would result in a greater score difference in a question. Thus, we measured the cosine similarity between subsequent submissions of a question. Specifically, we calculated cosine similarities between ith and (i-1) th submissions, for i => 2 attempts and took the average. We took all submissions because we wanted to assess how students' changed their answers upon receiving feedback, and how those changes impacted their final scores.

To encode students' responses into vector representations, we used the Universal Sentence Encoder (USE) [15]. The USE can take a word, a sentence, or a paragraph as inputs and encodes into a fixed length vector of 512 values. We then used a Deep Averaging Network (DAN) model with USE to encode questions and question-dependent texts into vectors [15]. DAN averages unigrams and bi-grams of word embedding to construct sentence embedding. Moreover, to evaluate how the answer modifications were connected to score differences, we calculated Spearman correlation between mean cosine similarities and score difference.

4.2 RQ2 Methodology

To answer RQ2, we coded student actions within the AL system as either an answer submission, reading event (R), or SRL event, such as annotating (A), highlighting (H), or vocabulary-lookup (V). The AL system does not define explicit student sessions.

Therefore, we adopted a data-driven approach from prior research to define session [36, 52]. First, we aggregated students' assignment actions and timestamps into a unified transaction log. We then plotted histograms of two consecutive action sequences to estimate the intervals between consecutive actions within an assignment. Based on our exploratory analysis we selected 30 minutes as a "session" cutoff. Any student actions exceeding 30 minutes were defined as a new session. After defining session cutoffs, we split all student actions within an assignment by session. Next, we counted SRL events ***before*** a student's resubmission of the question received feedback.

We then applied a four-level hierarchical linear model (HLM) to predict the last score of a question. HLM is commonly used in educational research [24, 50] to account for nested data [62]. Our HLM model included questions at level-one, assignment ID at level-two, student ID at level-three, and teacher ID at level-four. Fixed effect variables included students' first score on questions and features of SRL usage during attempting questions. All grouping variables were modelled as random intercepts.

4.3 RQ3 Methodology

To answer RQ3, we categorized teacher feedback comments using deductive analysis, which is a method for analyzing content using a predefined model based on prior research [23]. Our deductive analysis categories were adapted from Hattie and Timperley's [31] feedback categories, which have been used in prior research [1, 30]. Our model was also influenced by Shute's [53] review of formative feedback, and Bruno and Santos' [8] combined inductive and deductive coding scheme of teacher written comments in a science classroom context for task and processing-level feedback. We established five a-priori feedback categories using these models. The feedback categories included feedback on the task and processing [31] that (i) asked a student to either provide a correction to a response or to (ii) provide an explanation of a response [8], (iii) self-level feedback, (iv) and SRL behaviors. We also created a category for feedback that only addressed (v) conventions (e.g., spelling, grammar).

The SRL behavior category included teacher comments that referred to the SRL reading behaviors described in the previous section. SRL feedback has been defined as high-information feedback about task performance and suggestions for employing self-regulation strategies to monitor cognitive processes, self-evaluate performance, and strategy development to improve performance [31, 60]. We defined SRL feedback more broadly to include teacher comments that provided feedback on referring to the text to make revisions to an answer, annotating or highlighting the text, or to look up a vocabulary term. This definition is more appropriate within the context of AL, in which teachers leave brief comments on comprehension questions. Prior research has also defined SRL feedback in this context as feedback that includes knowledge about when to refer to the text [39] and which text information is relevant for completing the task [40].

Two members of the research team trained on coding comments using a sample. All differences in training were resolved by discussion. One researcher then coded all teacher feedback comments (n = 1,441). A second researcher independently coded 23% of this sample. Inter-rater reliability (IRR) was calculated using Cohen's kappa and was found to be acceptable ($\kappa = 0.74$, $p < 0.001$). We then applied a nonparametric Kruskal-Wallis test to identify if reading and SRL behaviors varied significantly among feedback categories.

5. RESULTS

In the following subsections we discuss our results for each RQ.

5.1 RQ1 Results

We calculated the average cosine similarities between subsequent submissions (sim_score) and score difference (d) with and without empty student submissions. A higher sim_score indicates that the submitted answers are more similar to each other. The frequencies of six different score difference categories and question counts (n) are: **-2** (n = 4), **-1** (n = 53), **0** (n = 187), **1** (n = 474), **2** (n = 252), **3** (n = 87), and **4** (n = 15). Total questions = 1,072. The Spearman correlation test between score difference (d) and mean cosine similarities (sim_score) was (coefficient = -0.315, $p < 0.001$). The negative coefficient indicates when the mean cosine similarity score decreases, the score difference increases. In other words, the more changes are present in students' subsequent answers, the greater the score difference.

Score Increased Descriptive statistics in this category are: 828 unique questions, 1,963 submissions by 543 students. First attempt score ranged from 0 to 3 with a mean 1.41. Last attempt score varied from 1 to 4 with a mean 2.98. We found that the positive score change groups have increased by 1, 2, 3, and 4 points. In these four groups, sim_score has a lower median value compared to the rest. This observation indicates that students with greater score increases had submissions that differed more than their original answers, as represented by a lower sim_score. We examined student submissions with identical responses (sim_score = 1) but an increase in final score (n = 40) submissions.

Score Decreased. This group includes 57 unique questions with 124 submissions by 49 students. First attempt scores ranged from 1 to 4 with a mean 1.58. Last attempt score varied from 0 to 3 with a mean of 0.51.

Score Unchanged. Descriptive statistics for this category are: 187 unique questions, 415 submissions by 165 students. First and last attempt scores have the same statistics in this category. First and last attempt scores ranged from 0 to 4 with a mean 1.69.

5.2 RQ2 Results

Standardized effect sizes were calculated using the formula, ß = (B*SDx)/(SDy) [50]. First attempt score had the highest predictive power (B = 0.32, ß = 0.28, $p < 0.001$). Only reading was a statistically significant positive predictor. Highlighting behavior was negatively associated with the last score.

5.3 RQ3 Results

Feedback comments were coded as requiring either a correction (n = 654), explanation (n = 565), a SRL behavior (n = 134), addressing conventions (n = 77), and self-level feedback (n = 11). SRL events after receiving feedback and before resubmission were identified. Kruskal-Wallis test results indicated statistically significant differences across the five feedback categories for reading ($p < 0.001$), highlighting ($p = 0.007$), and vocabulary lookup events ($p = 0.006$). Annotating text was not significant.

We conducted post-hoc analyses using Dunn's pairwise tests with Benjamini-Hochberg correction for features with statistically significant results. Effect size (r) is reported using a nonparametric test, Cliff's-Delta. Results indicated that students were less likely to engage in reading after conventions feedback when compared to SRL behavior feedback ($p < 0.001$, r = 0.25), corrective ($p < 0.001$, r = 0.28), and explanation feedback ($p < 0.001$, r = 0.29). The Kruskal-Wallis test assessed whether the group with non-zero entry (i.e., SRL Behavior) was statistically

different from the ones with all zero entries. We found statistically significant differences between SRL Behavior and corrective feedback (p = 0.002, r = 0.009) and explanation feedback (p = 0.001, r = 0.009). Students were more likely to look up vocabulary words after corrective over explanation feedback (p = 0.003, r = 0.021).

6. DISCUSSION and CONTRIBUTIONS

Scholarly Implication: Student Response to Feedback

RQ1 results show that students' who modified their answers had greater score differences, which is consistent with prior findings on automated feedback [39, 40, 64]. We found that teachers at times scored revised responses lower than students' initial score. When examining students' responses, we found students sometimes submitted the identical answer or an empty answer ("No response") despite the teacher asking for explanation or suggesting additional correction. This phenomenon in which students do not address teacher feedback is known as the ***"feedback gap"*** [24]. Students might not respond to feedback if they find it difficult to decipher [9], lack study habits [20], or erroneously believe it does not apply to them [28]. One limitation of the present study is that it is not equipped to determine the reason for lack of student response.

Our HLM analysis from RQ2 shows that reading events and initial scores were statistically significant predictors of last scores. However, SRL variables such as annotation, highlighting, and vocabulary lookups were not statistically significant predictors. We also found that highlighting was underutilized by students and that self-level feedback was not commonly employed by teachers.

Feedback comments categorized as focusing on correction, explanation, and SRL behaviors were associated with more reading events during student revisions when compared to feedback about conventions. We expected SRL feedback to produce more reading events and SRL behaviors than other categories based on prior research with automated feedback, because these comments directed students to revisit the text to revise their answers [39, 40]. However, SRL feedback did not produce statistically significant differences in student behaviors compared to correction and explanation feedback. One reason for this finding might be that these feedback categories had similar amounts of information; the level of feedback informativeness may have a greater impact on student performance and behavior [60]. Corrective (e.g., "Protons cannot be gained or lost") and explanation feedback provided did not explicitly direct students to revisit the text or use an AL feature, but perhaps these behaviors were implied perhaps these behaviors were implied during a task-oriented reading assignment with explanation feedback comments such as: "Great definitions but you need to explain why phase changes are considered physical changes.". This might explain why vocabulary look-ups were more common in corrective and explanation comments when compared to SRL (e.g., "Go back and reread paragraph 9 and reanswer. Might help you to plug some numbers into the equation to see how the inverse relationship works.") and conventions feedback ("Capitalize the first word in a sentence."). Conversely, perhaps the SRL feedback could more effectively influence reading events and SRL behaviors if teachers provided more explicit information that helped students decide when and how to revisit the text to revise an answer [39] or required students to select relevant information from the text to support their answer [40]. SRL feedback may have directed students to relevant portions of the text based on relatively greater highlighting behavior after SRL feedback, but this effect size was small, and highlighting was not positively related to score change, calling into question the value of this behavior.

For Teachers: Feedback Quality

Our analysis also showed that teacher comments were generally short and contained limited information. It may be possible for teachers to improve the quality and effectiveness of their feedback by providing more SRL feedback [60, 31], and by avoiding self-level feedback and comments about conventions, which were shown to not support student performance in the present study. To optimize feedback from teacher comments and increase student feedback uptake, teachers should support students in understanding feedback comments and evaluation criteria [13], which may require greater elaboration within comments and potentially instruction outside of AL.

Design Implications: Automated Feedback Affordances

Feedback can improve performance [38], but poor feedback can hinder student learning [31]. Middle school teachers may not have enough time to provide quality and timely feedback to all students, particularly when providing feedback to open-ended questions that require source-based explanations [6]. Although automated feedback may assist teachers in providing quality and timely feedback without increasing their workload [6, 14], challenges remain in building platforms to provide such feedback within the context of source-based science questions. For example, AL science questions are often constructed responses that require connecting information from different paragraphs. The state-of-the-art NLP research to automatically infer information from paragraphs in reading comprehension is still in early stage [22, 35]. One possible design for automated support would be to collect other teachers' feedback on the same question in the AL platform and provide suggestions to the teacher.

Design Implication: Supporting Feedback Actionability

Some students were not responsive to feedback as indicated by their submission of an empty answer or re-submission the same answer. One solution to increase actionability could be pointing to additional learning materials in an automated feedback setting [33]. For example, Broos and colleagues [7] designed a button "***Okay, what now?***" in a dashboard to provide actionable feedback. Students could click the button to view extra reading content. Similarly, a nudge can be implemented in AL—***"Are you sure you want to submit that empty answer?"***

7. CONCLUSIONS

This study has two main contributions to reading and SRL research: (i) empirically evaluating students' response changes to short answer questions upon receiving feedback and (ii) measuring the association of students' reading and SRL with five feedback categories. Our findings show that students who revised their answers demonstrated statistically significant differences in their scores. We also observed that teachers mainly provided corrective feedback followed by explanatory and feedback related to SRL behavior. Students exhibited more reading behavior upon receiving these types of feedback than convention-related feedback. These results may aid educators in writing feedback comments to students for maximal impact.

8. ACKNOWLEDGEMENTS

This research was supported by NSF #1821475 ``Concert: Coordinating Educational Interactions for Student Engagement'' Collin F. Lynch, Tiffany Barnes, and Sarah Heckman (Co-PIs).

9. REFERENCES

[1] Rola Ajjawi and David Boud. 2017. Researching feedback dialogue: An interactional analysis approach. Assessment & Evaluation in Higher Education, 42, 2, 252-265. DOI:https://doi.org/10.1080/02602938.2015.1102863

[2] Bita Akram, Wookhee Min, Eric Wiebe, Bradford Mott, Kristy Elizabeth Boyer, and James Lester. 2018. Improving Stealth Assessment in Game-Based Learning with LSTM-Based Analytics. International Educational Data Mining Society.

[3] Roger Azevedo. 2008. The role of self-regulated learning about science with hypermedia. Recent innovations in educational technology that facilitate student learning (2008): 127-156.

[4] L. P. Beaudoin, and P. Winne. 2009. nStudy: An Internet tool to support learning, collaboration and researching learning strategies. In Canadian e-Learning Conference

[5] Andrew Biemiller, and Naomi Slonim. 2001. Estimating root word vocabulary growth in normative and advantaged populations: Evidence for a common sequence of vocabulary acquisition. Journal of educational psychology 93, 3, 498.

[6] Anthony F. Botelho and Neil T. Heffernan. 2019. CROWDSOURCING CHAPTER 11–FEEDBACK TO SUPPORT TEACHERS AND STUDENTS. Design Recommendations for Intelligent Tutoring Systems: Volume 7-Self-Improving Systems, 101.

[7] Tom Broos, Laurie Peeters, Katrien Verbert, Carolien Van Soom, Greet Langie, and Tinne De Laet. 2017. Dashboard for Actionable Feedback on Learning Skills: Scalability and Usefulness. In Learning and Collaboration Technologies. Technology in Education (Lecture Notes in Computer Science), Panayiotis Zaphiris and Andri Ioannou (Eds.). Springer International Publishing, Vancouver, Canada, 229–241.

[8] Inês Bruno and Leonor Santos. 2010. Written comments as a form of feedback. Studies in Educational Evaluation, 36, 3, 111–120. DOI:http://dx.doi.org/10.1016/j.stueduc.2010.12.001

[9] Deirdre Burke. 2009. Strategies for using feedback students bring to higher education. Assessment & Evaluation in Higher Education 34, 1, 41-50.

[10] Deborah L. Butler, and Philip H. Winne. 1995. Feedback and self-regulated learning: A theoretical synthesis. Review of educational research 65, 3,: 245-281.

[11] Therese Bouffard-Bouchard, Sophie Parent, and Serge Larivee. 1991. Influence of self-efficacy on self-regulation and performance among junior and senior high-school age students. International journal of behavioral development 14, 2, 153-164.

[12] David Carless and David Boud. 2018. The development of student feedback literacy:enabling uptake of feedback. Assessment & Evaluation in Higher Education 43, 8 (2018), 1315–1325. https://doi.org/10.1080/02602938.2018.1463354

[13] David Carless and Naomi Winstone. 2020. Teacher feedback literacy and its interplay with student feedback literacy. Teaching in Higher Education (2020): 1-14.

[14] Anderson Pinheiro Cavalcanti, Arthur Diego, Rafael Ferreira Mello, Katerina Mangaroska, André Nascimento, Fred Freitas, and Dragan Gašević. 2020. How good is my feedback? a content analysis of written feedback. In Proceedings of the Tenth International Conference on Learning Analytics & Knowledge, 428-437.

[15] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant et al. 2018. Universal Sentence Encoder for English. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 169-174. 2018

[16] Richard Correnti, Lindsay Clare Matsumura, Laura Hamilton, and Elaine Wang. 2013. Assessing students' skills at writing analytically in response to texts. The Elementary School Journal, 114, 2, 142-177.

[17] Jennifer G. Cromley and Roger Azevedo. 2007. Testing and refining the direct and inferential mediation model of reading comprehension. Journal of Educational Psychology 99, 2, 311.

[18] Mihai Dascalu, Philippe Dessus, Ștefan Trausan-Matu, Maryse Bianco, and Aurélie Nardy. 2031. ReaderBench, an environment for analyzing text complexity and reading strategies. International Conference on Artificial Intelligence in Education, pp. 379-388. Springer, Berlin, Heidelberg, 2013.

[19] Mihai Dascalu, Larise L. Stavarache, Stefan Trausan-Matu, Philippe Dessus, Maryse Bianco, and Danielle S. McNamara. 2015. ReaderBench: An integrated tool supporting both individual and collaborative learning." In Proceedings of the fifth international conference on learning analytics and knowledge, pp. 436-437. 2015.

[20] Phillip Dawson, Michael Henderson, Tracii Ryan, Paige Mahoney, David Boud, Michael Phillips, and Elizabeth Molloy. 2018. Technology and feedback design. Learning, design, and technology.

[21] Project DRIVER_SEAT https://www.neilheffernan.net/projects/funded-projects/driver seat

[22] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1 (Long and Short Papers), pp. 2368-2378.

[23] Satu Elo and Helvi Kyngäs. 2008. Qualitative Research and Content Analysis. Journal of Advanced Nursing, 62, 107-115.

[24] Carol Evans. 2013. Making sense of assessment feedback in higher education. Review of educational research 83, 1, 70-120.

[25] Effat Farhana, Teomara Rutherford, and Collin F. Lynch. 2020. Associations Between Self-Regulated Learning Strategies and Science Assignment Score in a Digital Literacy Platform. International Society of the Learning Sciences (ISLS) .

[26] Actively Learn Feedback https://help.activelylearn.com/hc/en-us/articles/115000590154-Revise-answers

[27] Tomer Gal, and Arnon Hershkovitz. 2019. Different Types of Response-Based Feedback in Mathematics: The case of textual and symbolic messages. In Proceedings of the 9th

International Conference on Learning Analytics & Knowledge, pp. 265-269. 2019.

[28] Teresa Garcia. 1995. The Role of Motivational Strategies in Self-Regulated Learning. New Directions for Teaching and Learning 63, 29-42.

[29] Niki Gitinabard, Tiffany Barnes, Sarah Heckman, and Collin F. Lynch. 2019. What Will You Do Next? A Sequence Analysis on the Student Transitions Between Online Platforms in Blended Courses. International Educational Data Mining Society (2019).

[30] Marjan J.B. Govaerts , Margje W.J. van de Wiel , Cees P.M. van der Vleuten. 2013. Quality of Feedback following Performance Assessments: Does assessor expertise matter? European Journal of Training and Development, 37, 105-125. DOI: https://doi.org/10.1108/03090591311293310

[31] John Hattie and Helen Timperley. 2007. The power of feedback. Review of Educational Research, 77, 81-112. DOI:10.3102/00346543029848

[32] Anton Havnes, Kari Smith, Olga Dysthe, and Kristine Ludvigsen. 2012. Formative assessment and feedback: Making learning visible. Studies in Educational Evaluation 38, 21–27. DOI:http://dx.doi.org/10.1016/j.stueduc.2012.04.001

[33] Christothea Herodotou, Sarah Heiser, and Bart Rienties. 2017. Implementing randomised control trials in open and distance learning: a feasibility study. OpenLearning: The Journal of Open, Distance and e-Learning 32, 2, 147–162.

[34] Hamideh Iraj, Anthea Fudge, Margaret Faulkner, Abelardo Pardo, and Vitomir Kovanović. 2020. Understanding students' engagement with personalised feedback messages. In Proceedings of the Tenth International Conference on Learning Analytics & Knowledge, pp. 438-447. 2020.

[35] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. Transactions of the Association for Computational Linguistics 8 (2020): 64-77.

[36] Vitomir Kovanović, Dragan Gašević, Shane Dawson, Srećko Joksimović, Ryan S. Baker, and Marek Hatala. 2015. Penetrating the black box of time-on-task estimation. In Proceedings of the fifth international conference on learning analytics and knowledge, pp. 184-193. 2015.

[37] Franki YH Kung, and Abigail A. Scholer. 2018. Message Framing Influences Perceptions of Feedback (In) directness. Social Cognition 36, 6, 626-670.

[38] Philip Langer. 2011. The Use of Feedback in Education: A Complex Instructional Strategy. Psychological Reports 109, 3, (December 2011): 775–84.

[39] A. C. Llorens, R. Cerdán and Eduardo Vidal-Abarca. 2014. Adaptive formative feedback to improve strategic search decisions in task-oriented reading. Journal of Computer Assisted Learning, 30, 3, 233–251. DOI:http://dx.doi.org/10.1111/jcal.12050

[40] A. C. Llorens, Eduardo Vidal-Abarca and R. Cerdán. 2016. Formative feedback to transfer self-regulation of task-oriented reading strategies. Journal of Computer Assisted Learning, 32, 4, 314–331. DOI:http://dx.doi.org/10.1111/jcal.12134

[41] Tamas Makany, Jonathan Kemp, and Itiel E. Dror. 2009. Optimising the use of note‐taking as an external cognitive aid for increasing learning. British Journal of Educational Technology 40, 4, 619-635.

[42] Linda H. Mason and Laura R. Hedin. 2011. Reading science text: Challenges for students with learning disabilities and considerations for teachers. Learning Disabilities Research & Practice, 26, 4, 214–222. DOI:http://dx.doi.org/10.1111/j.1540-5826.2011.00342.x

[43] Wannisa Matcha, Dragan Gašević, Nora'Ayu Ahmad Uzir, Jelena Jovanović, and Abelardo Pardo. 2019. Analytics of learning strategies: Associations with academic performance and feedback. In Proceedings of the 9th International Conference on Learning Analytics & Knowledge, pp. 461-470.

[44] Danielle S. McNamara. 2004. SERT: Self-explanation reading training. Discourse processes 38, 1 , 1-30.

[45] Danielle S. McNamara., Irwin B. Levinstein, and Chutima Boonthum. 2004. START: Interactive strategy training for active reading and thinking. Behavior Research Methods, Instruments, & Computers 36, 2, 222-233.

[46] Danielle S. McNamara, Tenaha P. O'Reilly, Rachel M. Best, and Yasuhiro Ozuru. 2006. Improving adolescent students' reading comprehension with iSTART. Journal of Educational Computing Research 34, 2, 147-171.

[47] Danielle S. McNamara. 2007. Reading comprehension strategies: Theories, interventions, and technologies. Psychology Press., 2007.

[48] David Nicol. 2010. From monologue to dialogue: improving written feedback processes in mass higher education. Assessment & Evaluation in Higher Education 35, no. 5 (2010): 501-517.

[49] Paul R. Pintrich. 2000. The role of goal orientation in self-regulated learning. 2000. Handbook of self-regulation, 451-502. Academic Press, 2000

[50] Anna D. Rowe, and Leigh N. Wood. 2008. Student perceptions and preferences for feedback. Asian Social Science 4, 3,: 78-88.

[51] Teomara Rutherford. 2017. Within and between person associations of calibration and achievement. Contemporary Educational Psychology 49, 226-237.

[52] Adithya Sheshadri, Niki Gitinabard, Collin F. Lynch, Tiffany Barnes, and Sarah Heckman. 2018. Predicting student performance based on online study habits: a study of blended courses.International Educational Data Mining Society (2018).

[53] Valerie J. Shute. 2008. Focus on formative feedback. Review of Educational Research, 78, 153–189. DOI:http://dx.doi.org/10.3102/0034654307313795

[54] Tuncay Türkben. 2019. The Effect of Self-Regulation Based Strategic Reading Education on Comprehension, Motivation, and Self-Regulation Skills. International Journal of Progressive Education 15, 4,: 27-46.

[55] Melanie R Weaver. 2006. Do students value feedback? Student perceptions of tutors' written responses. Assessment & Evaluation in Higher Education 31, no. 3 (2006): 379-394.

[56] Philip H.Winne and Allyson F. Hadwin. 1998. Studying as self-regulated learning. The educational psychology series. Metacognition in educational theory and practice (277–304). Lawrence Erlbaum Associates Publishers.

[57] Philip H. Winne and Nancy E. Perry. 2000. Measuring

self-regulated learning. Handbook of self-regulation, 531-566. Academic Press, 2000.

[58] Philip H. Winne and Allyson F. Hadwin. 2012. The weave of motivation and self-regulated learning. 2012. Motivation and self-regulated learning , pp. 309-326. Routledge, 2012.

[59] Philip H. Winne , John Cale Nesbit, Ilana Ram, Zahia Marzouk, Jovita Vytasek, Donya Samadi, and Jason Stewart. 2017. Tracing Metacognition by Highlighting and Tagging to Predict Recall and Transfer. AERA Online Paper Repository (2017).

[60] Benedikt Wisniewski, Klaus Zierer, and John Hattie. 2020. The power of feedback revisited: A meta-analysis of educational feedback research. Frontiers in Psychology, 10, 1-14. DOI:http://dx.doi.org/10.3389/fpsyg.2019.03087

[61] Carol Withey. 2013. Feedback engagement: forcing feed-forward amongst law students. The Law Teacher 47, no. 3 (2013): 319-344.

[62] Heather Woltman, Andrea Feldstain, J. Christine MacKay, and Meredith Rocchi. 2012. An introduction to hierarchical linear modeling. Tutorials in quantitative methods for psychology 8, 1, 52-69.

[63] Larry D Yore. 2012. Science literacy for all: More than a slogan, logo, or rally flag!. In Issues and challenges in science education research, pp. 5-23. Springer, Dordrecht, 2012.

[64] Mengxiao Zhu, Hee-Sun Lee, Ting Wang, Ou Lydia Liu, Vinetha Belur, and Amy Pallant. 2017. Investigating the impact of automated feedback on students' scientific argumentation. International Journal of Science Education 39, 12, 1648-1668

[65] Barry J. Zimmerman, and Albert Bandura. 1994. Impact of self-regulatory influences on writing course attainment. American educational research journal 31, 4 (1994), 845-862.

[66] Barry J. Zimmermann. 1989. Models of self-regulated learning and academic achievement. Self-Regulated Learning and Academic Achievement: Theory, Research and Practice, 1-26.

[67] Barry J. Zimmermann. 1989. A social cognitive view of self-regulated academic learning. Journal of educational psychology, 81, 3 ,329.

[68] Barry J. Zimmerman. 2000. Self-efficacy: An essential motive to learn. Contemporary educational psychology, 25, 1, 82–91

[69] Barry J. Zimmerman. 2000. Attaining self-regulation: A social cognitive perspective. Handbook of self-regulation, 13-39. Academic Press, 2000.

Analysis of Factors Influencing User Contribution and Predicting Involvement of Users on Stack Overflow

Maliha Mahbub
University of Saskatchewan
Saskatoon, Canada
mam789@mail.usask.ca

Najia manjur
University of Saskatchewan
Saskatoon, Canada
nam907@mail.usask.ca

Mahjabin Alam
University of Saskatchewan
Saskatoon, Canada
natasha.mahjabin@gmail.com

Julita Vassileva
University of Saskatchewan
Saskatoon, Canada
jiv@cs.usask.ca

ABSTRACT

Active involvement of new community members is essential for Q&A platforms such as Stack Overflow, to make the platform efficient and more inclusive. However, more than half of Stack Overflow users contribute only once and disappear. This decreases the diversity of viewpoints and experience on the platform. This paper aims to identify factors that can discourage users from active participation after their first or second post. We collected a dataset of the responses to questions posted by new users (answers, comments, upvotes, downvotes) and analysed the tone of the feedback and its impact on the users' ongoing participation. We considered as new users those who registered to Stack Overflow for the last two years before the data collection and classified then into three groups based on the number of their posts (low, medium and high number of posts) on Stack Overflow. The differences in the responses between the three user groups have been validated by performing one-way ANOVA and Pearson's chi-square test. Based on these results we trained a machine learning model using a SVM classifier which predicts whether a user is likely to post or not with an accuracy of 88.69 %. Our work contributes to identifying and quantifying the potential underlying factors behind the decline in participation and dropout of new users on Stack Overflow.

Keywords

Stack Overflow, User Analysis, One-way ANOVA, SVM Classifier, User Prediction Model

1. INTRODUCTION

Stack Overflow (SO) is one of the most popular Q & A based platforms for programmers, having over 50 million monthly visitors[7], over 16 million questions[19] and 19 million answers[11]. SO has detailed guidelines for posting questions and some fundamental standards for providing feedback to the questions. Expert users who have been using Stack Overflow for a long time are rewarded with badges and reputation [14]. In order to garner a high reputation on the site, the user must be active on a regular basis on the SO platform and their questions must have many positive responses. However, novice users may not have the correct vocabulary or expertise to formulate a technical question. As a result, they may end up getting negative responses such as "stupid question" or "This is such common issue, just google". Such negative responses to posts may discourage users to limit or seize their contributions to the platform. This kind of users make up for almost half the users of the platform [18]. Unsolved questions in SO have seen an exponential growth over the years [16] and they continue to be an issue for new users who seek help. Studies have shown that online trolling and negative responses worsen over the active time of a user in a community [8] and 77% of users tend to ask for help on the SO platform only once [13]. In this paper, we identify and validate the factors which impact the state of participation of the new users on Stack Overflow. "New users" are those who have been registered to SO for less than 2 years until August 2019 (when the dataset was created). We selected and analysed five features from the SO post responses to understand how getting little to no response and negative responses is related to users posting behaviour. Based on our findings, we built a machine learning classifier to predict posting status of users in SO.

Maliha Mahbub, Najia Manjur, Mahjabin Alam and Julita Vassileva "Analysis of Factors Influencing User Contribution and Predicting Involvement of Users on Stack Overflow". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 827-832. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

2. RELATED WORK

The Stack Overflow or SO platform has turned into a valuable resource for both skilled and amateur programmers for glitches, bug and any code related problems. Managing such a large user base has been a challenge and an ongoing topic for investigation and research from different perspectives. Anderson et al. [3] explores the correlation between user reputation and quality of answers and its impact on the design of the site. Asaduzzaman et al. [4] mines the unanswered questions in SO to reveal the underlying factors that lead to questions remaining unanswered, such as title length, askers' score, post length etc. Alharthi et al. [2] investigates sev-

eral factors that impact the quality of questions in SO and predicts the score of the question, which indicates its overall quality. Similar idea of prediction has been explored in Shao et al. [17] developed a prediction model which analyses the latent context of a question and recommends an answer for the user. Calefato et al. [6] developed a framework based on successful questions on SO to provide an evidence -based guideline for programmers to write better questions in SO. Grant et al. [12] explores the use of badge, to motivate users. When a question has better wording and quality, it attracts more users and the user gets upvotes which in turn helps the score. Adaji et al. [1] investigates specific social support strategies that influence users to contribute in SO More recent studies have focused on the behavioural and personality traits of SO users as well, in order to target the emotional aspects of the users who ask questions [15, 5]. The novice or infrequent users who just started out face some level of criticism or neglect by the more experienced users on SO, a phenomenon related to maintaining community boundaries by hazing. Hazing is a psycho-social phenomenon where the newcomers in a tightly knit group face backlash and elitist attitude (which is sometimes borderline abusive)[9]. Slag et al. [18] discusses the difficulties encountered by "one day flies", users who post only once in their profile's lifetime and do not contribute to the platform afterwards. Our work further investigates the effect of the factors identified in [18] by providing statistical and empirical validation to the hypothesis proposed in [18].

3. RESEARCH QUESTIONS AND DATASET

Since we wanted to compare the responses of the posts, not the nature of post itself, we eliminated two of Slag et al. [18] factors: duplicate questions and uncommon tags. Instead, to further investigate the features of questions asked by such inactive users, we added five new factors: the number of upvotes on a question (Up Votes), the downvotes (Down Votes), the number of comments on a post (Comment Count), the reputation of users (Reputation), and the types of comments on a post (Comment Texts). We chose to add reputation since it affects how a user's posts is perceived by other users. We aim to answer two research questions:
"1. Do these factors have any quantifiable relation to the frequency of posts of users in Stack Overflow?"
"2. Can we predict whether a user will drop out and stop posting?".

3.1 Data Collection

We collected data from Stack Exchange Data Explorer [1], an open source tool to collect publicly available data from Stack Overflow. We used Stack Exchange Data Explorer to collect information about users who created their profile on Stack Overflow in 2017.

For our work, we chose to consider only the questions posted by users as their contribution. We collected the number of answers, comments, upvotes, downvotes, view count given against (received by) each post of a user and the user's reputation. We decided to analyze the mean values of these features for each user so that we can consider all of them in a normalized form since the distribution of responses is not equal for all users. In order to determine the overall tone of a comment, we inferred the polarity of the comments from the text of the comments through sentiment analysis on the text by using TextBlob on all the comments received by a user. Polarity generally falls within the range of -1 to 1 where -1 refers to a very negative sentiment, 0 refers to a neutral sentiment and 1 refers to a very positive sentiment. We categorized this range into five different groups as shown in Table 1.

[1]https://data.stackexchange.com/stackoverflow/query/new

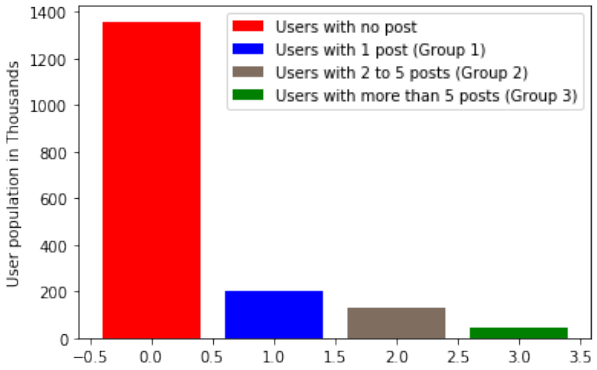


Figure 1: Distribution of users population (in thousands) based on the number of posts they made on Stack Overflow

Table 1: Group based on overall polarity of the comment

Group	Description
1 (Highly negative)	If -1 <= polarity <-0.5
2 (Moderately negative)	If -0.5 <= polarity <0
3 (Neutral)	0
4 (Moderately positive)	If 0 <polarity <= 0.5
5 (Highly positive)	If 0.5 <polarity <= 1

3.2 Target user groups

The users were categorized into the following groups based on the number of questions they posted from January 2017 to June 2019: group 1 (users who posted a question once), group 2 (users who have posted from 2 to 5 times), group 3 (users who have posted more than 5 times). The distribution of user population according to the number of posts they made in their lifetime on SO in out collected data is depicted in the figure 1. We aim to identify such users by analyzing their past experiences on SO. In our prediction model, described in Section 5, for predicting the future behavior of a user, group 1 is labelled as the negative class whereas groups 2 and 3 are combined into a single category as the positive class. Therefore, the labeled classes considered for this study are:

1. Negative class: Users who will discontinue making any contribution to Stack Overflow after their first post.

2. Positive class: Users who will continue contributing to the platform.

4. DATA ANALYSIS

	View Count	AnswerCount	CommentCount	Reputation	DownVote	UpVote	Polarity
View Count	1	0.32	0.41	0.1	0.3	0.51	0.24
AnswerCount	0.32	1	**0.86**	0.1	**0.59**	**0.68**	**0.67**
CommentCount	0.41	**0.86**	1	0.11	**0.57**	**0.61**	**0.79**
Reputation	0.1	0.1	0.11	1	0.08	0.24	0.1
DownVote	0.3	**0.59**	**0.57**	0.08	1	**0.52**	0.53
UpVote	**0.51**	**0.68**	**0.61**	0.24	**0.52**	1	0.49
Polarity	0.24	**0.67**	**0.79**	0.1	**0.53**	0.49	1

Figure 2: Distribution of users population (in thousands) based on the number of posts they made on Stack Overflow

The data analysis of feature selection, validation of metrics and prediction model is provided below.

4.1 Feature Selection

From the data collected from SO, we performed Pearson correlation analysis to find out how strongly each feature is related to another. The results of correlation analysis for the features are shown in figure 2. Following the Cohen's classification system [10], only the largest relationships i.e. where the correlation coefficient r>0.5, have been considered to be significantly correlated. From figure 2, it is evident that the correlation coefficient of comment count, answer count, downvote, upvote, and polarity are significant i.e. greater than 0.5. Therefore, for our feature analysis, these five features are selected as the final metrics for next stage.

4.2 Feature Distribution in User Groups

To answer the research question: Do these features have any quantifiable relation to the frequency of posts of users in Stack Overflow?, we analyzed their statistical differences among the three user groups. We used a one-way ANOVA test and Pearson's chi-square test to establish the statistical evidence of the differences in terms of the features among the three user groups.

All of the five features are plotted against the number of posts from users. And the plotting is done for each of the three target user groups to observe the difference of plots in each groups. The sections below provide in-depth description of each of the features on all three user groups.

4.2.1 Number of Answers against Number of Posts

Figure 3 depicts the distribution of average number of answers against the number of posts from each user from the target group of users on SO. The mean number of answers among three groups are: 1.10 (group 1), 3.31 (group 2) and 15.70 (group 3), which indicates that users in group 3 receive significantly more responses to their posts compared to users in groups 1 and 2. The p-value in one-way ANOVA test indicates significant statistical difference among the three groups in terms of the mean number of answers they receive against their posts ($F(2,375196) = 678.8$, $p = .000$).

4.2.2 Number of Comments against Number of Posts

The mean number of comments among three groups are: 2.22 (group 1), 6.60 (group 2) and 30.00 (group 3), which indicates that the mean number of comments significantly increases with the increasing number of posts in each group. Moreover, it is also evident from figure 4 that users who posted less (no more than five times) exhibited a higher tendency of receiving no comments from other users. The result of one-way ANOVA test indicates significant statistical difference between the groups than within the groups in terms of number of comments they receive against their posts ($F(2,375196) = 187.3$, $p = .000$).

4.2.3 Number of Upvotes against Number of Posts

The graphs show the relation between the average number of upvotes with the number of posts in Figure 5. The mean upvotes in group 1 and 2 are significantly lower than group 3 (0.696, 1.853 and 8.877 respectively), which indicates that the posts made by the users of group 3 are more appreciated and receive higher number of upvotes than the posts made by the users who are less active. One-way ANOVA result indicates significant statistical difference among the three groups in terms of number of upvotes they receive against their posts ($F(2,375196) = 17.15$, $p = .000$)

4.2.4 Number of Downvotes against Number of Posts

From figure 6, it can be observed that the mean number of downvotes in group 1 and group 2 (0.548 and 1.309 respectively) are lower than that of group 3 (4.17). This means that the users who are posting more questions are also getting fewer downvotes. This is an important and surprising observation since a higher number of posts could have also led to increased number of downvotes, which turns out to not be the case. One-way ANOVA result indicates significant statistical difference between the groups than within the groups in terms of number of comments they receive against their posts ($F(2,375196) = 473.04$, $p = .000$).

4.2.5 Comment Polarity against Number of Posts

Table 2: Percentage of each comment polarity category received by the user groups

	User Group		
Polarity Group	1	2	3
1 (Highly negative)	0.2%	0.1%	0%
2 (Moderately negative)	20.1%	20.7%	13.6%
3 (Neutral)	24.4%	11.7%	1.3%
4 (Moderately positive)	48.4%	67.2%	85.1%
5 (Highly positive)	1.1%	0.3%	0%

The result of cross tabulation in Table 2 revealed that users from group 3 received zero highly negative comments, and the least proportion of moderately negative comments. They also received the highest proportion of moderately positive comments (85.1%). On the other hand, the users of group 1 received the lowest amount of moderately positive comments(49.5%) among the user groups. It can be concluded from Table 2 that with the increase in the number of posts made by the users, there is an increase in the positive comments and decline in the negative remarks received by the post owners. Lastly, the result of Pearson's chi-square test establishes the statistically significant relationship between the polarity of comments and the user groups ($\chi^2(8, 297447) = 20310.29, p = 0.000$).

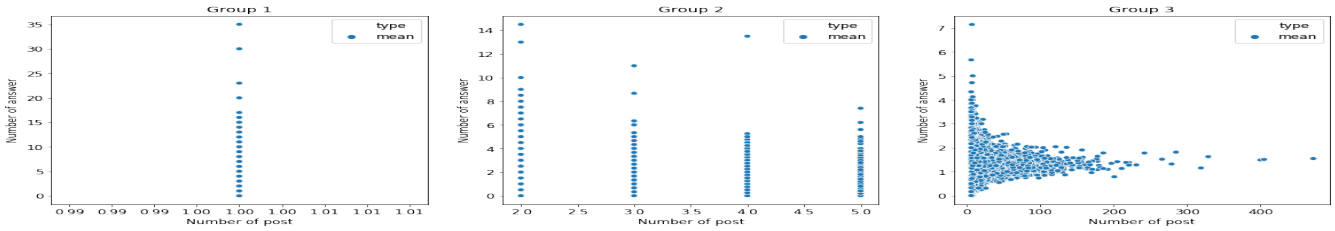


Figure 3: Distribution of average number of answers against number of posts among three user groups

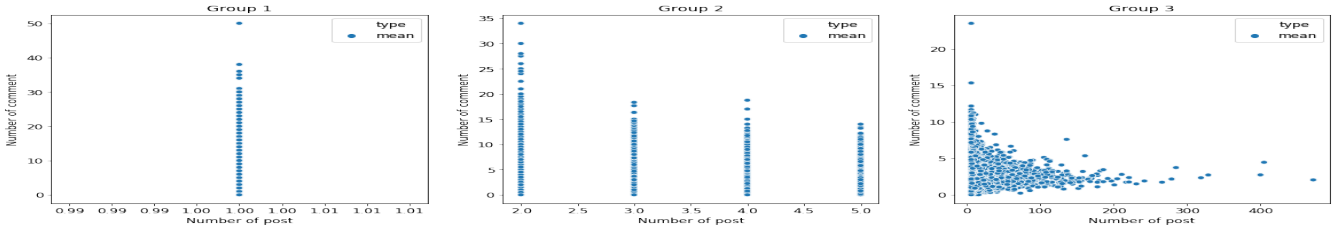


Figure 4: Distribution of average number of comments against number of posts among three user groups

	Name	Measure
Features	Mean answer	Scale
	Mean comment	Scale
	Mean upvote	Scale
	Mean downvote	Scale
	Mean polarity	Scale
Target	User Class	Nominal

Table 3: Attributes of the dataset employed in the SVM classifier

Classifier	Accuracy	Precision	Recall
SVM	0.8869	0.9875	0.7635

Table 4: Prediction model performance per evaluation metric

5. PREDICTING USER PARTICIPATION

Based on the findings from correlation analysis and statistical testing of factors influencing users post frequency, we developed an actual prediction model to answer the second research question. In order to develop and train our model, we took advantage of a popular supervised machine learning algorithm called support vector machines (SVM). Table 3 describes the features and target groups employed in our model. We divided the original data set into training set representing 80% of the data and testing set representing the remaining data. By using the five features, we divide our users into two classes: likely to post and not likely to post. The performance of our prediction model i.e. how well it predicts the user class is evaluated using the metrics: accuracy, precision and recall. Our model performs significantly well and yields a high score in terms of all three metrics as illustrated by Table 4.

6. IMPLICATIONS AND FUTURE WORK

From our data analysis, we observed that a low number of answers and comments, a high number of downvotes and negative comments, and a low number of upvotes are more prevalent in the posts of users who have posted fewer times compared to the users who have higher number of posts. Since these users are receiving negative remarks and downvotes even with fewer posts, this may play a role in discouraging them from seeking help again from SO. In the light of this discovery, we trained a SVM classifier model with the five special features and divided the users into two classes: users who will post in the future and users who will not. The model has shown good performance with high accuracy and effectiveness.

Previous works which mostly focused on how users can ask better questions or build a better profile to attract more answers to their questions. The novelty of our study is to identify infrequent users and find a possible factor underlying their withdrawal, so that the community owner/ moderator can make the platform more welcoming and less hostile for them. Our study has some limitations as well. We could not consider the number of deleted questions of a user as one of the factors that could contribute to users' decline in posts since Stack Exchange Data Explorer does not provide that data. The research also lacks a qualitative analysis from feedback of infrequent or absent users. Therefore, as part of our our future plan, we will attempt to explore the user modelling of infrequent posting through a targeted qualitative user study of SO users.

7. CONCLUSIONS

More than half of the users in Stack Overflow tend to ask for help on the platform only once and never post again. In this paper, we identified five main features / metrics which we hypothesized to be related to the inactive status of users. We collected the responses to posts in SO for users who have their SO profiles for 2 years (2017 to 2019) and selected five factors with strong correlation. Ours statistical analysis supports our hypotheses and validates the effect of these factors having a significant correspondence to users' posting frequency. Using these factors as selected features, we trained a machine learning model that predicts whether or not a user will post in the Stack Overflow platform, based

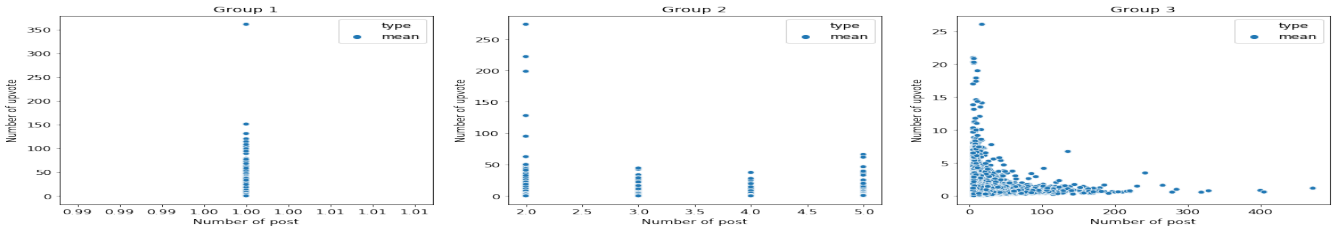


Figure 5: Distribution of average number of upvotes against number of posts among three user groups

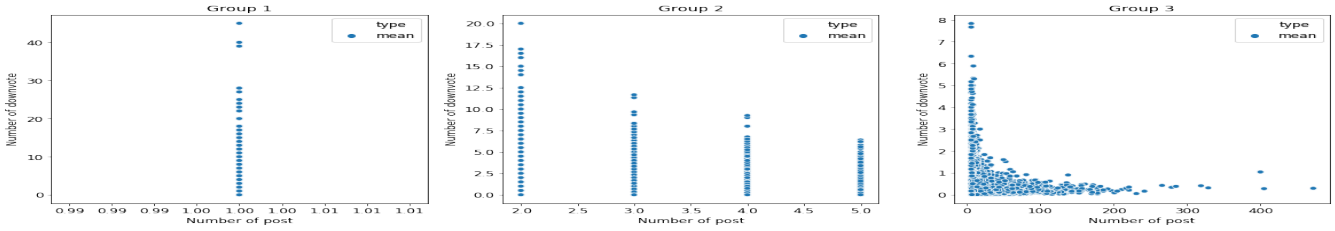


Figure 6: Distribution of average number of downvotes against number of posts among three user groups

on the responses their posts received till now. This prediction can identify users who have reduced their posting in SO and face lack of encouragement and thus can benefit from a positive nudge, help or mentorship. The significance of the contribution of our analysis and prediction model is that it can help to provide more equitable treatment of newcomers, and thus increase the diversity of the SO community.

8. REFERENCES

[1] I. Adaji and J. Vassileva. Towards understanding users' motivation in a q&a social network using social influence and the moderation by culture. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 349–350, 2017.

[2] H. Alharthi, D. Outioua, and O. Baysal. Predicting questions' scores on stack overflow. In *2016 IEEE/ACM 3rd International Workshop on CrowdSourcing in Software Engineering (CSI-SE)*, pages 1–7. IEEE, 2016.

[3] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 850–858, 2012.

[4] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider. Answering questions about unanswered questions of stack overflow. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 97–100. IEEE, 2013.

[5] B. Bazelli, A. Hindle, and E. Stroulia. On the personality traits of stackoverflow users. In *2013 IEEE international conference on software maintenance*, pages 460–463. IEEE, 2013.

[6] F. Calefato, F. Lanubile, and N. Novielli. How to ask for technical help? evidence-based guidelines for writing questions on stack overflow. *Information and Software Technology*, 94:186–207, 2018.

[7] P. Chatterjee, M. Kong, and L. Pollock. Finding help with programming errors: An exploratory study of novice software engineers' focus in stack overflow posts. *Journal of Systems and Software*, 159:110454, 2020.

[8] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec. Antisocial behavior in online discussion communities. In *Ninth International AAAI Conference on Web and Social Media*, 2015.

[9] A. Cimino. The evolution of hazing: Motivational mechanisms and the abuse of newcomers. *Journal of Cognition and Culture*, 11(3-4):241–267, 2011.

[10] J. Cohen. Statistical power analysis. *Current directions in psychological science*, 1(3):98–101, 1992.

[11] Craig Smith. Interesting stack overflow statistics and facts (2020). https://expandedramblings.com/index.php/stack-overflow-statistics-and-facts/, 2020. [Online; accessed 25-March-2020].

[12] S. Grant and B. Betts. Encouraging user behaviour with achievements: an empirical study. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 65–68. IEEE, 2013.

[13] John Slegers. The decline of stack overflow. https://hackernoon.com/the-decline-of-stack-overflow-7cb69faa575d, 2015. [Online; accessed 26-March-2020].

[14] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Steenkiste, and C. Faloutsos. Analysis of the reputation system and user contributions on a question answering website: Stackoverflow. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 886–893. IEEE, 2013.

[15] N. Novielli, F. Calefato, and F. Lanubile. Towards discovering the role of emotions in stack overflow. In *Proceedings of the 6th international workshop on social software engineering*, pages 33–36, 2014.

[16] M. M. Rahman and C. K. Roy. An insight into the unresolved questions at stack overflow. In *2015*

IEEE/ACM 12th Working Conference on Mining Software Repositories, pages 426–429, May 2015.

[17] B. Shao and J. Yan. Recommending answerers for stack overflow with lda model. In *Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*, pages 80–86, 2017.

[18] R. Slag, M. de Waard, and A. Bacchelli. One-day flies on stackoverflow-why the vast majority of stackoverflow users only posts once. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 458–461. IEEE, 2015.

[19] Wikipedia contributors. Stack overflow — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Stack$_Overflowoldid = 940723804, 2020.[Online; accessed20 - February - 2020].

SimGrade: Using Code Similarity Measures for More Accurate Human Grading

Sonja Johnson-Yu, Nicholas Bowman, Mehran Sahami, and Chris Piech
Stanford University
{sonja, nbowman, sahami, piech}@cs.stanford.edu

ABSTRACT

While the use of programming problems on exams is a common form of summative assessment in CS courses, grading such exam problems can be a difficult and inconsistent process. Through an analysis of historical grading patterns we show that inaccurate and inconsistent grading of free-response programming problems is widespread in CS1 courses. These inconsistencies necessitate the development of methods to ensure more fairer and more accurate grading. In subsequent analysis of this historical exam data we demonstrate that graders are able to more accurately assign a score to a student submission when they have previously seen another submission similar to it. As a result, we hypothesize that we can improve exam grading accuracy by ensuring that each submission that a grader sees is similar to at least one submission they have previously seen. We propose several algorithms for (1) assigning student submissions to graders, and (2) ordering submissions to maximize the probability that a grader has previously seen a similar solution, leveraging distributed representations of student code in order to measure similarity between submissions. Finally, we demonstrate in simulation that these algorithms achieve higher grading accuracy than the current standard random assignment process used for grading.

Keywords

similarity, code embeddings, embeddings, assessment, grading, human, simgrade, grade

1. INTRODUCTION

Free-response coding questions are a common component of many exams and assessments in programming courses. These questions are popular because they give students the opportunity to show their understanding of course material and demonstrate their coding and problem-solving skills [16]. However, the flexible nature of these problems introduces unique challenges when it comes to grading student responses, which are compounded in situations where the scale of the course necessitates a team of graders working together ("group grading"). The difficulty of consistent application of grading criteria by a group of graders stems from the incredible diversity of student submissions that are generated for free-response coding questions. In particular, it has been previously shown that the space of different student solutions to free-response programming problems follows a long-tailed Zipf distribution [18]. For this reason, it is challenging to develop automated systems for grading and providing feedback and thus human grading remains the gold standard for grading such free-response problems. However, even a team of human graders with extensive experience can struggle to consistently and accurately apply a single, unified criteria when grading. This is problematic as it can result in negative impacts on students in the form of incorrectly assigned grades and inaccurate feedback. Our goal in this paper to explore the frontier of techniques improving the process and outcomes of the exam grading experience.

Our main insight in developing improved approaches for grading is that *it is easier for graders to grade in a consistent manner if they are able to grade similar submissions one after another.* First, we examine historical data to provide concrete evidence of a relationship between grader accuracy and the similarity of previously graded submissions to the current submissions a grader is grading. Then, we propose algorithms that group and order similar submissions in different ways to minimize grader error. Finally, we show that these algorithms perform better than current baseline methods for grading. This work's primary contributions are:

1. Reporting of grader errors in a CS1 course
2. Using historical data to demonstrate the potential benefits of similarity-based grading
3. Three algorithms for grading using code similarity

1.1 Related Work

Autograding One commonly used approach to scale grading is the use of autograders [6]. While useful for comparing program output for correctness or matching short snippets of code, autograders are more problematic for free-response questions in exam settings. In such contexts, the subtlety of understanding that human graders provide is often essential to providing appropriate feedback to students and properly assessing the (partial) correctness of their solutions. While promising, fully autonomous AI solutions are not ready for

Sonja Johnson-Yu, Nicholas Bowman, Mehran Sahami and Chris Piech "SimGrade: Using Code Similarity Measures for More Accurate Human Grading". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 833-837. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

grading CS1 midterms [14, 11, 18, 12] especially for contexts with only hundreds of available student submissions [17].

Grading by Similarity The idea of grouping and organizing student submissions in order to improve grading outcomes has been previously proposed for a variety of problem types. Merceron and Yacef [9] use vectors that encode students' mistakes in order to group together students who make similar mistakes when working on formal proofs in propositional logic. Gradescope, designed by Singh et al. [15] offers functionality for grading similar solutions, which is currently most effective on multiple-choice-type questions. This approach has also been applied to short answer questions, as explored by Basu et al. [2], as well as math problems, as demonstrated by Mathematical Language Processing [8]. In this paper, we identify "similar" student responses on free-response programming questions to improve grading quality.

Code Similarity In order to define similarity metrics for student code submissions, we apply techniques for generating numerical embeddings for student programs. Henkel et al.what [5] created abstracted symbolic traces, a higher-level, light-syntax summary of the programs, and embedded them using the GloVe algorithm [13]. Alon et al. [1] pioneered code2vec, an attention-based embedding model specifically used to represent code. Recently, further advances have been made to improve code embeddings by training contextual AI models on large datasets from Github [7]. For this application, we favor simpler unsupervised embedding strategies that do not require human-generated labels by adapting the popular NLP technique Word2vec [10], in which "word" representations are derived from surrounding context.

1.2 Dataset

Our analysis focuses on the student submissions and grader logs from four exams for an introductory programming (CS1) course taught in Python. The breakdown of summary statistics across the four exams is presented in Table 1. As a note, a "submission" is defined as one student's written answer to one free-response problem – thus, the total number of submissions for a given exam is roughly the number of students times the number of coding problems on the exam. In total, we analyze 11,171 student submissions across 1,490 students. Additionally, we have grading logs for every student submission, which consists of information about the grader, the criteria items applied, the final score, and the amount of time that the grader spent on the submission. 199 graders contributed to grading these four exams. As discussed below, the same student submission is sometimes graded by more than one grader for validation purposes. Thus, our dataset contains 14,597 individual grading log entries.

Our grading data comes from a grading software system that randomly distributes student submissions to graders. Among the standard student submissions for grading, this software also inserts "validation" submissions that have already been graded by senior teaching assistants. Every grader assigned to a specific problem will grade all "validation" submissions for that problem. The presence of these special submissions creates opportunities for assessing grader performance, both relative to their peers and relative to "expert" performance.

Exam #	# Students	# Submissions	# Graders
1	533	3,731	53
2	259	1,813	52
3	247	2,470	51
4	451	3,157	43
Total	1,490	11,171	199

Table 1: Exam Grading Dataset Summary Statistics

2. NATURAL GRADING ERROR

While anecdotal experience of grading inconsistency is a common trend in our experience as educators, our first focus is to quantify the inconsistencies present in historical grading sessions in a rigorous manner. In particular, our analysis focuses on the aforementioned "validation" submissions that were specially handled by the grading software and assigned to every grader working on a specific problem. As a result, we had a subset of the grading logs for which we knew both the true grade (as defined by an expert) and the "validation" grade assigned by each grader. Plotting these values against one another is shown in Figure 1, which reveals troubling inconsistencies in the grades assigned by graders. With an RMSE of 7.5 (i.e., average error of 7.5 percentage points per problem), we see that grading error is significant, nearly on the order of what would translate to a full letter grade. Linear regression on this plot yields an R-squared coefficient of 0.947 indicating that while the error may be high, the direction of errors is generally unbiased. In other words, there is not systematic over/under-grading. Rather, the grading errors tend to be randomly distributed around the true grade. Thus, the rest of this paper focuses on methods for decreasing this demonstrated inconsistency (absolute error) in human grading.

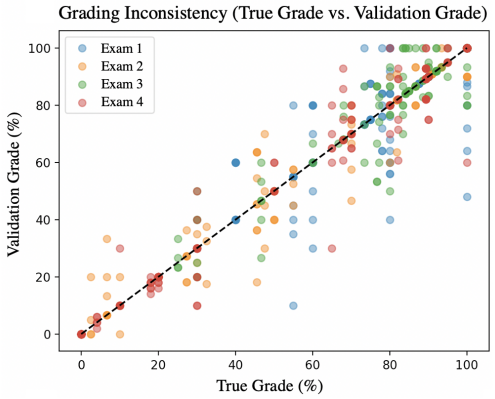


Figure 1: True grade assigned by expert vs. validation grade assigned by human grader

3. METHODS

In this section, we will first outline methods for answering key questions about the problem of improving human grading using similarity scores. Then, we will present three novel algorithms for improving human grading.

3.1 Can code similarity be accurately captured?

We generate program embeddings for all student submissions in our corpus. Word embeddings are an established

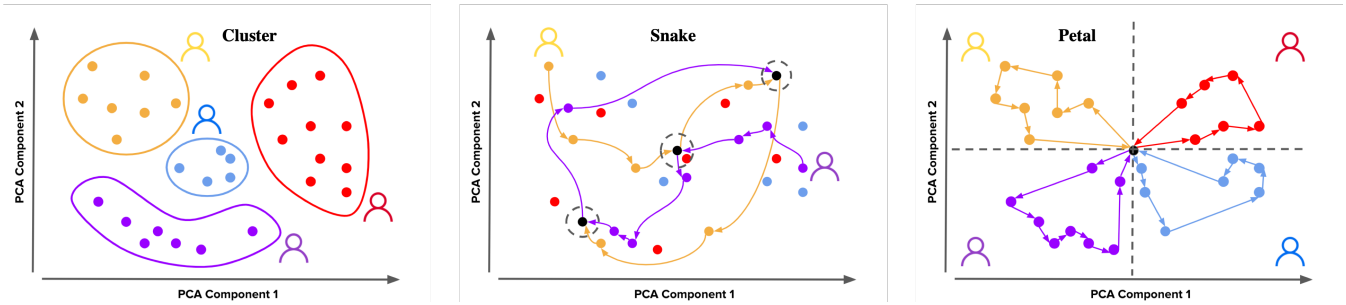


Figure 2: Submission assignment via three algorithms: Cluster, Snake, Petal

method of encoding semantics in human language [10, 13, 3, 3], and these same techniques applied to code accomplish similar results. Algorithms for generating embeddings are constantly evolving and improving; to avoid over-optimization at the embedding generation stage, we chose to employ the simple baseline Word2Vec algorithm. We then demonstrated that our embeddings are semantically significant using zero-shot rubric sampling [18]. For details, see the Appendix[1].

3.2 Does similarity influence grader accuracy?

We hypothesize that graders score submissions more accurately when they have recently seen a submission similar to the current submission. To test this hypothesis, we analyze grading data for four exams. First, for each grader, we generate a "percentage grading error," which is an average of their absolute percent deviation from the correct answer on all validation submissions that they graded. Then, for each of the validation submissions that a grader evaluated, we sort their personal grading logs by time and look at the window of three submissions leading up to each validation submission they graded. To quantify similarity of the validation submission to recently graded submissions, we take the maximum of the cosine similarity between the current validation submission and the three previous submissions. We plot the maximum similarity between a validation submission and the previous submissions against a grader's percentage grading error in order to identify the relationship between a grader's history and accuracy. Then we can infer a formula that approximates the relationship between previous submission similarity and percentage grading error.

3.3 Algorithms to assist human grading

We compare four algorithms for assigning submissions to graders: (1) Random, in which submissions are randomly assigned to graders, with five "validation" submissions interspersed for assessing grader bias. This is the status quo and serves as the baseline. (2) Cluster, in which each grader is assigned to a cluster of highly similar submissions. (3) Snake, in which each grader is randomly assigned a set of submissions and is shown the submissions greedily by nearest neighbor. (4) Petal, in which the dataset is divided into "petals" and all graders begin in the same place. Figure 2 provides a visualization of (2), (3), and (4). Detailed explanations of the algorithms are in the Appendix[1].

[1]https://compedu.stanford.edu/papers/appendices/SimGradeAppendix.pdf

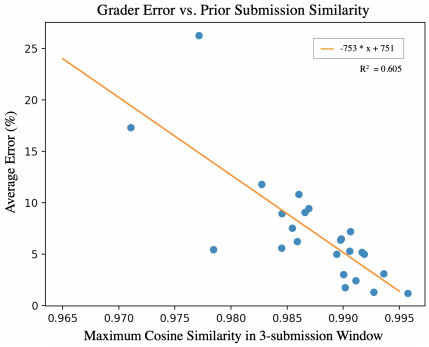


Figure 3: Relationship between grader accuracy and similarity in 3-submission window prior to validation submission

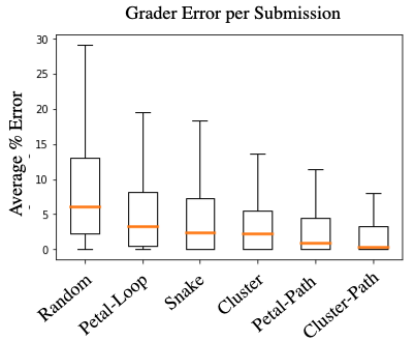
3.4 Algorithm evaluation

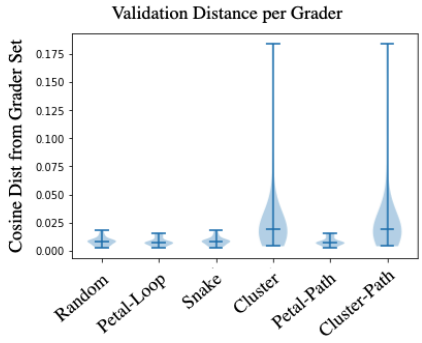
To evaluate the performance of the different algorithms, we simulate grading for a 444-person six-problem exam and ten graders, using real student programs from an actual exam. Details about the selection of validation submissions are in the Appendix[1]. When running the simulation, we infer percentage grading error by examining the similarity of the previous three submissions to the current submission. While we emphasize grader error as the most important metric for assessing an algorithm, a secondary consideration is how naturally validation submissions integrate with the rest of a grader's assigned submissions. Ideally, a validation submission is not "out-of-distribution" with respect to the other submissions that a grader is assigned. Otherwise, a grader will be able to tell when they are being evaluated for grading accuracy. To assess how "out-of-distribution" the validation submissions are, we examine how dissimilar the validation submissions are from the non-validation submissions assigned to a grader. Specifically, for each validation submission, we measure the distance between the validation submission and the nearest non-validation submission assigned to that grader. We average over the five validation submissions in order to get the mean minimum distance from validation to non-validation for a grader, which will be higher if one of the validation submissions is out-of-distribution.

4. EXPERIMENTAL RESULTS

4.1 Similarity scores are meaningful

Embeddings are semantically significant because similarity between embeddings corresponds to similarity between sub-





Algorithm	Grading Error (%)	Validation Dist (cos)	P-Value
Random	10.2	0.0084	-
Petal-Loop	6.1	0.0080	0.07
Snake	6.2	**0.0068**	0.10
Cluster	3.9	0.028	< 0.005
Petal-Path	3.9	0.0080	< 0.01
Cluster-Path	**2.7**	0.028	< 0.005

Figure 4: Left: Average per-submission grading error for each algorithm, Center: Distance of validation submissions from normally assigned submissions, Right: Summary performance statistics, including comparison to random baseline.

mission feedback labels, as described in the Appendix[1].

4.2 Similarity influences grading

Graders score assignments more accurately when they have recently seen a submission similar to the current submission they are grading. From our analysis of historical data, we find that when there is a high similarity between the current submission and at least one of the previous three submissions, the percentage grading error is low. Conversely, when the similarity between previous submissions is low, the percentage grading error is high. We find a linear relationship between the maximum similarity of the previous three submissions and the percentage grading error as shown in Fig. 3, with $R^2 = 0.605$. Given that the grading process involves the numerous uncertainties that come along with human involvement, we believe this correlation coefficient shows a statistically significant relationship between historical similarity and grader accuracy. While the linear relationship between historical submission similarity and percentage grading error is a simplifying assumption, it is the best assumption we can make given evidence provided in Fig. 3.

4.3 Improved accuracy by algorithm

We compare six algorithms for assigning submissions to graders and selecting an order in which a grader will view a submission in Figure 4. We apply the equation of the linear relationship shown in Figure 3 to the similarity of submissions as ordered for evaluation by different algorithms in our experiments. This equation allows us to predict grader accuracy when using the orderings provided by different algorithms. We find that implementing a path ordering on a clustered assignment of graders to submissions yields the lowest mean error of 2.7% (bold-ed in Fig. 4), while the other algorithms all show an improvement over the baseline 10.2% grading error. We utilize bootstrapping [4] over 100,000 trials in order to get the p-values that indicate the significance of the difference in means between the baseline algorithm and the other algorithms (see table in Fig. 4).

4.4 Validation viability by algorithm

When comparing the cluster, snake, and petal algorithms, we observe that the cluster-based algorithms are most likely to have validation submissions that are "out-of-distribution," with a mean validation distance of 0.0277. All other algorithms have substantially lower mean minimum distances.

5. DISCUSSION

Overall, we saw that all of our novel proposed algorithms for assignment of submissions to graders provided improvements over the random baseline in simulation. In general, we saw that path-based algorithms (petal-path and cluster-path) had lower grading error than their non-path counterparts because they are designed to optimize for maximum similarity between consecutive submissions that a grader grades. In particular, the cluster-path algorithm yielded the lowest grader error in simulation due to its strong tendency to assign very similar submissions to graders. On the other hand, the snake algorithm provided the most optimal average distance to validation submissions, which may be important for a smooth experience for a real-life grader. Finally, we saw that the petal algorithm offered a balanced trade-off between these two extremes – while not optimal in either metric, it can be a good choice when both metrics (grading error and validation submission distance) are equally important for designing a grading experience. For a more in-depth discussion of our observed results, see the Appendix[1].

6. CONCLUSION

Through analysis of historical exams, we demonstrated that there is inconsistency between true scores and grader-assigned scores. In doing so, we introduce a new task and associated measure, *grading correctness*. Moreover, we found experimental support for our hypothesis that graders are able to assign scores to exam problems more accurately when they have previously seen similar submissions. In turn, we proposed the use of code embeddings to capture semantic information about the structure and output of programs and identify similarity between submissions. Using similarity of code embeddings in conjunction with historical grading data, we demonstrate in simulation that graders are indeed able to score submissions more accurately when they have previously seen another submission similar to it. We propose and compare several algorithms for this task, showing that it is possible to achieve a significant increase in grading accuracy over simple random assignment of submissions. Future extensions of this work include (i) improvements on code embeddings and (ii) deployment of the grading algorithms in an operational system to allow more direct experimental comparison of grading accuracy. The use of such algorithms show promise for improving accuracy, and in turn fairness, in evaluations of student performance.

7. REFERENCES

[1] U. Alon, M. Zilberstein, O. Levy, and E. Yahav. code2vec: Learning distributed representations of code. *CoRR*, abs/1803.09473, 2018.

[2] S. Basu, C. Jacobs, and L. Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the ACL*, October 2013.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[4] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, Jan. 1979.

[5] J. Henkel, S. Lahiri, B. Liblit, and T. W. Reps. Code vectors: Understanding programs through embedded abstracted symbolic traces. *CoRR*, abs/1803.06686, 2018.

[6] M. Joy, N. Griffiths, and R. Boyatt. The boss online submission and assessment system. *J. Educ. Resour. Comput.*, 5(3):2–es, Sept. 2005.

[7] A. Kanade, P. Maniatis, G. Balakrishnan, and K. Shi. Learning and evaluating contextual embedding of source code, 2019.

[8] A. S. Lan, D. Vats, A. E. Waters, and R. G. Baraniuk. Mathematical language processing: Automatic grading and feedback for open response mathematical questions. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, L@S '15, pages 167–176, New York, NY, USA, 2015. ACM.

[9] A. Merceron and K. Yucef. Clustering students to help evaluate learning. *Technology Enhanced Learning*, 2004.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[11] A. Nguyen, C. Piech, J. Huang, and L. Guibas. Codewebs: scalable homework search for massive open online programming courses. In *Proceedings of the 23rd international conference on World wide web*, pages 491–502, 2014.

[12] S. Parihar, Z. Dadachanji, P. K. Singh, R. Das, A. Karkare, and A. Bhattacharya. Automatic grading and feedback using program repair for introductory programming courses. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 92–97, 2017.

[13] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[14] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas. Learning program embeddings to propagate feedback on student code, 2015.

[15] A. Singh, S. Karayev, K. Gutowski, and P. Abbeel. Gradescope: A fast, flexible, and fair system for scalable assessment of handwritten work. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, L@S '17, pages 81–88, New York, NY, USA, 2017. ACM.

[16] D. Thissen, H. Wainer, and X.-B. Wang. Are tests comprising both multiple-choice and free-response items necessarily less unidimensional than multiple-choice tests?an analysis of two tests. *Journal of Educational Measurement*, 31(2):113–123, 1994.

[17] K. Wang, B. Lin, B. Rettig, P. Pardi, and R. Singh. Data-driven feedback generator for online programing courses. In *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*, pages 257–260, 2017.

[18] M. Wu, M. Mosse, N. Goodman, and C. Piech. Zero shot learning for code education: Rubric sampling with deep learning inference, 2018.

A Time-Aware Approach to Detect Patterns and Predict Help-Seeking Behaviour in Adaptive Educational Systems

Raquel Horta-Bartomeu
Universidad Nacional de Educación a Distancia (UNED)
rhorta3@alumno.uned.es

Olga C. Santos
aDeNu Research group
Universidad Nacional de Educación a Distancia (UNED)
ocsantos@dia.uned.es

ABSTRACT

In distance education and some computer-assisted learning scenarios asking for help when needed is important. Some students do not ask for help even when they do not know how to proceed. In situations where a teacher is not present, this can be a serious setback. We aim to find an approach to learn about students' help-seeking behaviour by studying sequences of actions that end with the student asking for help. The goal is to be able to recognize those students who need help but fail to ask for it and offer them assistance. We propose to include the temporal context of user-platform interaction and suggest an ensemble model to learn from both general and personal tendencies.

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education; I.2.6 [**Artificial Intelligence**]: Learning; I.5.4 [**Pattern recognition**]: Applications; G.3 [**Probability and Statistics**]: Markov processes, Time series analysis

Keywords

Adaptive systems, time series, educational data mining, personalized education

1. INTRODUCTION

Researchers have found help-seeking to be important in learning scenarios and observed that some students do not reach for help when they need it [4, 25]. When a teacher is not always present, and the student needs some level of self-discipline, not asking for help might be problematic as the student could end up wheel-spinning [18] or abandoning the task. The longitudinal nature of student-platform interactions leads us to think that taking into account the temporal context could be useful for analysing help-seeking behaviour. There is literature on help-seeking including temporal data, and some works have focused on performance prediction or have centred on specific knowledge topics. However, we have not found work that focuses on the behaviour around help-seeking actions, including temporal data and independent of student knowledge and task content. In this Master Thesis, we propose to represent student-platform interactions as sequences of actions, study whether sequential patterns exist in students' help-seeking behaviour and explore whether a prediction model could identify students that need help but do not ask for it.

Raquel Horta-Bartomeu and Olga C. Santos "A Time-Aware Approach to Detect Patterns and Predict Help-Seeking Behaviour in Adaptive Educational Systems". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 838-843. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

2. RELATED RESEARCH

Time series studies are very common in natural sciences and some social sciences. Studies that make use of time series data can also be found in the field of educational sciences [5, 8, 9, 13, 14, 15, 17, 29]. We have reviewed existing works on both help-seeking behaviour and time series data analysis. In section 3 we highlight the specific differences between the works exposed here and what we propose to do.

2.1 Help-seeking behaviour

Knowing when to ask for help is important [4, 11]. [10, 11] agreed that it could improve resilience and efficacy. According to [10], help-seeking has been studied for years but the rise of new technologies opens new research opportunities on help-seeking in these new contexts.

Some works have focused on detecting specific situations that are known to be problematic. For instance, in classes where the teacher has more students than desired, it might be difficult for them to identify students who need help or are stuck. [18] developed a method using machine learning (ML) models to automatically predict wheel-spinning and decide how to intervene. [4] attacked both problems of asking for help too much and not enough by negotiating with the student. Rather than using ML models, they predefined a set of heuristics. A slightly different situation was studied by [32]. Their goal was to find a connection between student procrastination (i.e. intentionally delaying work) and their activities within different learning materials. They used data from a massive open online course (MOOC) platform and found two main study strategies: students who delayed work worked intensively for short periods followed by long pauses, while students who did not delay usually split the tasks into subtasks and worked more constantly but less intensively.

Other works have focused on knowledge tracing [6, 7, 24], however, we will not be considering student knowledge but their behaviour and interaction with the educational system.

2.2 Pattern recognition and sequence prediction

To cluster categorical sequences, one needs to define the function or method used to compare the sequences pairwise, that is, how to measure the distance between them. [5] analyzed activity frequency through the length of different online courses to study if different activity patterns were related to student performance. They used agglomerative hierarchical clustering (AHC) with the Levenshtein distance. [17] used a similar approach and found patterns in group problem-solving strategies analysing group behaviour of students working on interactive tabletops. [8] also used AHC with the Levenshtein distance and found 3 groups of similar study state sequences using data from a drill-and-practice learning environment in college mathematics.

[13] found that a large subgroup of MOOC participants might have been engaging by watching video lectures without doing the assignments. Their methodology consisted in constructing, for each student, vectors of states representing their engagement trajectories through the course. They computed the distance between trajectories by assigning a numerical value to each label and calculating the L1 norm.

[14] proposed a method that would capture the clusters' number and size evolution over time. They transformed log data sequences into Markov chain models. Then, they computed the pairwise similarities by computing the expected transition probabilities using the stationary distribution over the actions. They used the Jensen-Shannon divergence and the Hellinger distance between the expected transition frequencies of the Markov chains (more details in [21], as cited in [14]). They used k-means with an evolutionary clustering method that tracks the evolution of the similarities over time by smoothing the similarity matrices ([31] as cited in [14]). [9] also modelled student behaviour using Markov chains. They randomly generated Markov chain priors and assigned each sequence to the prior most likely to generate it. Then, each prior would be updated to the Markov chain generated using its associated sequences. These last two steps were repeated until less than 5% of the sequences would change their prior. As they stated, this method is similar to k-means but with the clustering being dependent on the Markov chains instead of on a similarity measure performed directly on the sequences.

[29] was able to detect unprofitable learning experiences and predict student performance by using time series data. They used dynamic time warping (DTW) to measure the distance between sequences and performed hierarchical clustering to find clusters. DTW was proved to be useful; however, to the best of our knowledge, it is not suitable for categorical sequences but only for numerical ones and has therefore been ruled out as a possible approach to our specific problem.

Artificial neural networks (ANNs) are known to be useful for a vast variety of tasks. When it comes to time series, the most used ones seem to be recurrent neural networks (RNN) and long short-term memory networks (LSTM). RNNs' main limitation is their difficulty to work with long sequences due to a vanishing gradient problem [12]. While LSTMs solve this issue, they usually take quite a long to train and have difficulties in capturing long-term dependencies in long sequences [12, 30]. Finally, a novel approach called transformer networks was introduced by [30]. This approach introduces what the authors called an *attention mechanism*, which solves the long-term dependency problem in LSTMs. [2] used LSTMs in a multi-module system to analyse the relationship between intent and user actions in interactive systems. [16] used time-aware LSTMs (T-LSTM), a special type of LSTM that can handle time irregularities, to model student knowledge state in continuous time. They conducted an empirical experiment and discovered that they outperformed regular LSTMs, logistic regression and recent temporal pattern mining (RTPs). [15] used RTPs along with support vector machine (SVM) and logistic regression to predict student performance and detect the need for intervention using students' answers to programming exercises. They were able to classify students within only 1 minute into the exercise.

3. EXPECTED CONTRIBUTION

To our knowledge, this would be the first work to use student-platform interaction data in form of sequences of actions to predict help-seeking behaviour while being independent of the topic being taught. Our approach differs from existing work by joining three main aspects. First, **help-seeking behaviour**: we have found works that linked the need for intervention with performance or student knowledge [6, 15, 16] instead of analyzing the behaviour surrounding actual help-seeking actions. Second, **time-awareness**: we have found works that have used cumulative data (e.g. number of attempts) to predict the need for intervention [18] but did not take into account the temporal context. Third, **topic-independence**: we have found works that did take into account the temporal context but focused on the content of student answers for specific topics [15, 23].

The approach we propose would include the temporal context, would not be dependent on the nature of the content being taught and would focus on user-platform interaction (i.e. clicking, typing, deleting, consulting theory, etc).

While this research is still in the early stages, we believe in the importance of students' affective state [26, 27, 28] and might consider including the affective context if possible. Finally, if we were to find successful results, we believe that richer predictions could be obtained by joining student knowledge information [6, 15, 16] along with the information learnt from help-seeking behaviour. However, this is out of the scope of this research.

4. RESEARCH QUESTIONS

We aim to study whether our proposal would be feasible and for that we present two research questions:

Q1 Are there temporal patterns in students help-seeking behaviour?

Q2 Can temporal student-platform interaction data be used to detect students who need help but do not ask for it?

5. PROPOSED METHODOLOGY

We will be dealing with both supervised and unsupervised problems: we will be using clustering algorithms towards answering Q1 and prediction algorithms towards answering

Q2. We propose to perform clustering (Q1) as a preliminary step to a more complex system (Q2). Clustering can lead to interpretable results and reveal information that could be useful to pedagogical experts while some prediction methods are more powerful but may act as a black box. As well as considering less interpretable methods, the system proposed in Q2 addresses personalization. We expose the methodology we intend to follow, and the methods we have considered so far.

5.1 Data

The dataset to be used is yet to be found or constructed. Efforts are being made to find a suitable dataset. Some promising options are being considered but are yet to be confirmed. Even though, the characteristics that we look for in a dataset have been defined. The dataset should contain action logs that originated from the interaction between a student and a learning platform that has some kind of help tool that the student can choose to use. Each log should include, at least: (1) action type, (2) action start time, (3) action end time, (4) student identification (anonymized) and (4) exercise identification.

Given that some actions are continuous rather than instantaneous, we will need to decide how to represent this characteristic. As an example, a student might consult the theory section of the system just for 10 seconds, or they could spend 5 minutes consulting the content. It would be desirable that those two cases were not represented in the same way and that duration was taken into account. When using Markov chains, if we consider action durations, the probabilities of staying in the same state will always be 0, and the duration would not be taken into account. To solve this, we could consider splitting the actions into time slots. We will need to take into account that some other actions might be instant actions, with practically no duration, e.g. submitting an exercise. We will need to make sure that the model we use does not undermine these actions. Finally, if possible, we might consider including *idle* actions, that is, time in which the student does nothing.

5.2 Clustering

To answer **Q1**, we encounter two main decisions: how to determine the distance between sequences and which clustering algorithm to use.

5.2.1 Distance between the sequences

The main challenge of dealing with sequential data is that they cannot be directly fed to traditional clustering algorithms. First, one needs to decide how to represent the sequences and define how to compare them. We have decided to try two methods for representing the distance between sequences: Markov chains and the Levenshtein distance. Markov chains represent a sequence by considering the probability of going from one state (i.e. action) to another. The basic form of a Markov chain only considers the current state to predict the next one. This could be a limitation and therefore n-order Markov chains could be considered. In a Markov chain of order n, n previous steps are taken into account. On the other hand, the Levenshtein distance is a type of edit distance, that is, the minimum changes required to transform one sequence into another. The Levenshtein distance considers insertions, deletions and substitutions.

5.2.2 Clustering algorithms

Taking into account existing work, we have narrowed the search for a clustering method down to two: hierarchical clustering and k-means. The main drawback of k-means is the requirement of a predefined number of clusters, which in our case is unknown. Hierarchical clustering has the advantage that the number of clusters can be chosen a posteriori, however, it can be expensive when dealing with large datasets. K-means is usually a fast algorithm, although it might depend on the chosen distance metric [19].

5.3 Prediction

Towards answering **Q2**, we propose a prediction system; its characteristics are presented in this section.

5.3.1 System structure

While we want to take advantage of how students in general behave, we want to provide a personalized learning experience. To do so, a student's personal traits and tendencies must be taken into account. Therefore, we aim to take advantage of the general traits of student behaviour while preserving the personal study tendencies of each student, thus combining an inter-subject with an intra-subject approach. To achieve this goal, we propose an ensemble system composed of three blocks. We name the system SHEmblE (Sequence analysis of Help-seeking behaviour with an ensEMBLE model for Educational systems)

Firstly, we will have a prediction model that will be trained with all the available data. We will refer to this model as the *common model* as it will be shared among all students. We expect it to be able to learn the general patterns of help-seeking behaviour if those exist.

Secondly, we will have what we call a *personal model*. Each student will have each own personal model trained with their own data, if any. We expect this model to be able to learn the personal tendencies and preferences of a student.

Finally, a third model will combine the predictions of common and personal models. We call this model the *ensemble model* and we expect it to learn how to combine the predictions the best way possible.

We will focus on students who regularly ask for help for the general model. However, from those students, we will take into account interactions that exhibit help-seeking behaviour as well as those in which the student does not need help to successfully reach their goal. Sequences from students who never ask for help will not be included as we cannot know if the student did really not need assistance, or they simply never ask for it.

We are aware that data size will be a concern regarding the personal model. Its goal is to provide individualization, and thus, we believe it is an important part of the system [1, 7, 22]. Therefore, the ensemble model could take into account the amount of data with which the personal model was trained in order to weigh the predictions properly.

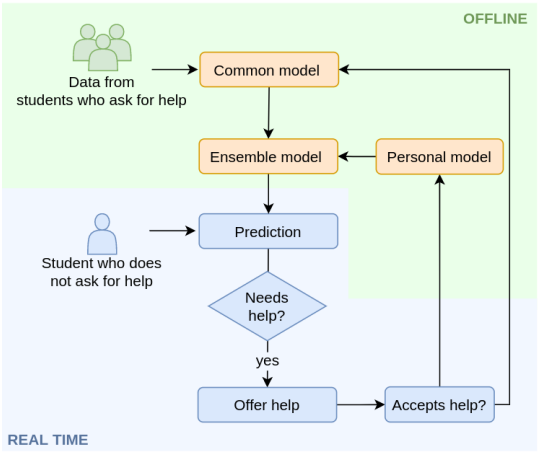


Figure 1: General schema of SHEmblE, the proposed system.

The ultimate goal, if this system was to achieve good results, would be to implement it in a real educational system. Figure 1 represents the overall structure of the proposed system. The idea would be to be able to detect, in real-time, students that need help and offer it to them. Apart from collecting logs from students that ask for help themselves, whenever we offer help we would save their response as well. The scope of this work comprises the common, personal and ensemble models, the rest could be the object of study of future research.

5.3.2 *Prediction algorithms*

Time series prediction has been the challenge of many works in literature. This work deals with categorical time series, in other words, categorical sequences. It has been narrowed down to three methods: artificial neural networks, hidden Markov models, and recent temporal patterns. As exposed in section 2, ANNs have been used in problems involving time series data and showed promising results, different types found in the literature will be considered (e.g. LSTM, T-LSTM, transformers). HMMs have also been useful for predicting and classifying action sequences. [20] found that HMMs needed fewer training samples and less CPU time while performing similar to LSTMs. Finally, RTPs [3] have been successful at similar tasks. [15] used them and managed to detect students that needed intervention only one minute after starting an exercise. While their data consisted of attributes of the students' answers' content and ours will consist of interaction data, we believe that a similar approach could be applied to our particular task.

5.3.3 *System training and evaluation*

The system we propose is going to be composed of three different models. These models will be evaluated independently and altogether. We intend to evaluate the common model by performing a variation of the leave-one-out cross-validation (LOOCV) in which in each iteration the whole data of one student is left out for validation. Regarding the personal model, the dataset will be split by the sequences' student id and for each student, a LOOCV will be performed. The performance of the personal model will be assessed by combining all the performances (eg. mean and standard deviation) and special attention will be paid to possible outliers. Finally, the ensemble model block will need to be fed the predictions of the other two models. Therefore, a whole new dataset E will need to be constructed such that:

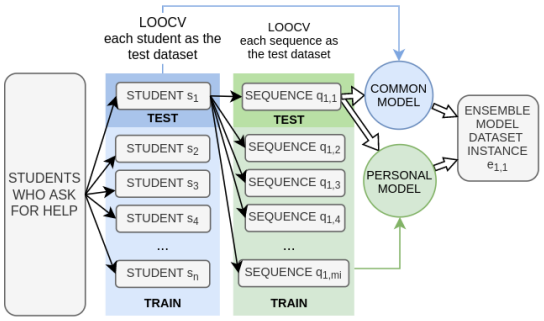


Figure 2: Graphical representation of the evaluation scheme and the generation of the dataset for the ensemble model.

- Consider the set of n students $S = \{s_i | i \in \{1..n\}\}$.
- Each student s_i has got m_i sequences $Q_i = \{q_{ij} | j \in \{1..m_i\}\}$
- The instance e_{ij} will correspond to the sequence j of the student i and will contain at least 2 features:
  - The output of the common model trained using the sequences from students other than s_i.
  - The output by the personal model trained using the sequences of student s_i other than q_{ij}.

  Moreover, additional features could be added, such as the size of the dataset used to train the personal model, given that some students might have few or no data.
- The dataset E will then have $\sum_{i=1}^{n} m_i$ rows.

Once the dataset has been constructed, k-fold cross-validation can be performed. Figure 2 shows a graphical representation of the proposed evaluation method.

6. CONCLUSIONS

We have not found works that aim to detect students who need help by analysing behaviour around help-seeking actions using time-aware user-platform interaction data. In this Master Thesis, we aim to study whether such data can be useful to predict help-request actions and propose an ensemble system that combines a shared model and a personal model so as to achieve individualization.

This work is still at a very early stage. Any feedback and ideas on this proposal are very much welcomed. Specifically, comments on the sequence representation, and the clustering and predictive model choices will be appreciated.

7. ACKNOWLEDGEMENTS

The work is partially supported by UNED's master's degree in Research in AI and the project INT2AFF funded under Grant PGC2018-102279-B-I00 (MCIU/AEI/FEDER, UE) by the Spanish Ministry of Science, Innovation and Universities, the Spanish Agency of Research and the European Regional Development Fund (ERDF).

8. REFERENCES

[1] M. Abell. Individualizing learning using intelligent technology and universally designed curriculum. *The journal of technology, learning and assessment*, 5(3), 2006.

[2] R. Agrawal, A. Habeeb, and C. Hsueh. Learning user intent from action sequences on interactive systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*, page 59–64, New Orleans, Louisiana, USA, February 2-7 2018.

[3] I. Batal, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht. Mining recent temporal patterns for event detection in multivariate time series data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 280–288, 2012.

[4] C.-Y. Chou, K. R. Lai, P.-Y. Chao, S.-F. Tseng, and T.-Y. Liao. A negotiation-based adaptive learning system for regulating help-seeking behaviors. *Computers & Education*, 126:115–128, 2018. ID: 271849.

[5] R. Conijn and M. V. Zaanen. Trends in student behavior in online courses. In *3rd International Conference on Higher Education Advances*, pages 649–657, 2017.

[6] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[7] A. T. Corbett, J. R. Anderson, V. H. Carver, and S. A. Brancolini. Individual differences and predictive validity in student modeling. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, 1994.

[8] M. Desmarais and F. Lemieux. Clustering and visualizing study state sequences. In *Proceedings of the 6th International Conference on Educational Data Mining (EDM 2013)*, pages 224–227, 2013.

[9] C. Hansen, C. Hansen, N. O. D. Hjuler, S. Alstrup, and C. Lioma. Sequence modeling for analysing student interaction with educational systems. In *Proceedings of the 10th International Conference on Educational Data Mining, EDM 2017*, pages 232–237. International Educational Data Mining Society (IEDMS), 2017.

[10] S. Järvelä. How does help seeking help?–new prospects in a variety of contexts. *Learning and Instruction*, 21(2):297–299, 2011.

[11] S. A. Karabenick and R. S. Newman. *Help Seeking in Academic Settings: Goals, Groups, and Contexts.* Lawrence Erlbaum Associates, Inc, 2006.

[12] F. Karim, S. Majumdar, H. Darabi, and S. Chen. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.

[13] R. Kizilcec, C. Piech, and E. Schneider. Deconstructing disengagement. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*, LAK '13, pages 170–179. ACM, 2013.

[14] S. Klingler, T. Käser, B. Solenthaler, and M. Gross. Temporally coherent clustering of student data. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM 2016)*, pages 202–209, 2016.

[15] Y. Mao. One minute is enough: Early prediction of student success and event-level difficulty during novice programming tasks. In *Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019)*, pages 119–128, 2019.

[16] Y. Mao, S. Marwan, T. W. Price, T. Barnes, and M. Chi. What time is it? student modeling needs to know. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, pages 171–182, 2020.

[17] R. Martinez, K. Yacef, J. Kay, A. Al-Qaraghuli, and A. Kharrufa. Analysing frequent sequential patterns of collaborative learning activity around an interactive tabletop. In *4th International Conference on Educational Data Mining, EDM 2011*, pages 111–120. CEUR-WS, 2011.

[18] T. Mu, A. Jetten, and E. Brunskill. Towards suggesting actionable interventions for wheel-spinning students. In *13th International Conference on Educational Data Mining (EDM 2020)*, pages 183–193, 2020.

[19] E. Ofitserov, V. Tsvetkov, and V. Nazarov. Soft edit distance for differentiable comparison of symbolic sequences. 2019. Preprint available at https://arxiv.org/abs/1904.12562.

[20] M. Panzner and P. Cimiano. Comparing hidden markov models and long short term memory neural networks for learning action representations. In *International Workshop on Machine Learning, Optimization, and Big Data*, pages 94–105. Springer, 2016.

[21] L. Pardo. *Statistical inference based on divergence measures*. Chapman and Hall / CRC Press, 2018.

[22] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 255–266. Springer, 2010.

[23] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas. Learning program embeddings to propagate feedback on student code. In *International conference on machine Learning*, pages 1093–1102. PMLR, 2015.

[24] C. Piech, J. Spencer, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.

[25] I. Roll, R. S. d. Baker, V. Aleven, and K. R. Koedinger. On the benefits of seeking (and avoiding) help in online problem-solving environments. *Journal of the Learning Sciences*, 23(4):537–560, 2014.

[26] S. Salmeron-Majadas, R. S. Baker, O. C. Santos, and J. G. Boticario. A machine learning approach to leverage individual keyboard and mouse interaction behavior from multiple users in real-world learning scenarios. *IEEE Access*, 6:39154–39179, 2018.

[27] S. Salmeron-Majadas, O. C. Santos, and J. G. Boticario. An evaluation of mouse and keyboard interaction indicators towards non-intrusive and low cost affective modeling in an educational context.

Procedia Computer Science, 35:691–700, 2014.

[28] O. C. Santos. *Emotions and personality in adaptive e-learning systems: an affective computing perspective*, pages 263–285. Emotions and personality in personalized services. Springer, 2016.

[29] S. Shen and M. Chi. Clustering student sequential trajectories using dynamic time warping. In *Proceedings of the 10th International Conference on Educational Data Mining (EDM 2017)*, pages 266–271, 2017.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[31] K. S. Xu, M. Kliger, and A. O. H. Iii. Adaptive evolutionary clustering. *Data Mining and Knowledge Discovery*, 28(2):304–336, 2014.

[32] M. Yao, S. Sahebi, and R. F. Behnagh. Analyzing student procrastination in moocs: A multivariate hawkes approach. In *13th International Conference on Educational Data Mining (EDM 2020)*, pages 280–291, 2020.

Mixed Data Sampling in Learning Analytics

Julian Langenhagen
Goethe University Frankfurt, Germany
langenhagen@econ.uni-frankfurt.de

ABSTRACT

Technical progress facilitates collecting large amounts and new kinds of data in a wide range of areas. That enables versatile new possibilities in empirical research, especially with high-frequency data. However, researchers are confronted with the problem that not all available data have the same (high) frequency. For many common methods, it is necessary to adjust the high-frequency to the low-frequency data, resulting in a significant loss of information. In accounting research, this kind of problem exists due to the low-frequency reporting data of companies on the one hand and the high-frequency financial market data on the other. A promising solution to this problem is the innovative approach of mixed data sampling (MIDAS). Since the coexistence of low-frequency data (e.g., exam grades) and high-frequency data (e.g., learning management system usage data) is also prevalent in educational settings, this paper will discuss the first application of MIDAS in the field of learning analytics.

Keywords

time series, mixed data sampling, regression, prediction models

1. INTRODUCTION

Educational Data Mining (EDM) is a comparatively young field of research. Even though certain research methods within this area are already established, the field regularly benefits from valuable contributions from interdisciplinary research approaches [e.g., 14]. The methods used in Educational Data Mining can be divided into four different areas: prediction models, structure discovery, relationship mining, and discovery with models [2]. The focus of this paper lies on prediction models, especially on those with time-series data. Methods in this area include classifications, regressions, and latent knowledge estimation [2]. In this area, EDM researchers are confronted with a specific problem. The variety of available data sources is growing due to the use of complex learning management systems (LMS), game-based learning applications, or other digital educational tools. These sources often contain high-frequency data and therefore offer a lot of potential information. However, in the most commonly used methods in prediction models, it is usually the case that data from different samples have to be brought to the same frequency to be analyzed. This can lead to a significant loss of information. For example, data from an LMS can be gathered for every possible usage second to be used as an explanatory variable. However, there is usually a low-frequency variable on the other side of the equation, such as the exam grade or score. Accounting researchers face a similar problem. Companies usually only publish reports at a pre-defined low frequency, as financial statements or other reports are often made available only annually or quarterly. An independent variable available in corresponding research settings is, for example, the share price, which, like the data from the LMS, can be collected at a very high frequency. However, the information contained here cannot be fully included in the analysis if the variables on both sides of the equation have to be adjusted to the lowest frequency available. A solution to this problem in accounting is the method of mixed data sampling (MIDAS). As the underlying problem is comparable to typical educational research settings, this method will be examined in more detail in the following section, and then a possible application in Educational Data Mining will be discussed using the example of a concrete implementation in the context of a learning app in higher education.

Julian Langenhagen "Mixed Data Sampling in Learning Analytics". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 844-846. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

2. MIDAS IN ACCOUNTING RESEARCH

The basic MIDAS model builds on a regression equation where the dependent variable is measured in a lower frequency than one or more of the independent variables [5]. The problem of the different frequencies on both sides is solved with two separate components. In the first component, each time disaggregated observation of the higher frequency variable is included separately as an independent variable. In other words, if the higher frequency data is observable N times within a period, then N separate independent variables are included in the regression. This allows the independent variable's effect on the dependent variable to evolve over the course of the examined period, even though the dependent variable was only measured once. The second component of MIDAS is the requirement of each of the N regression coefficients to follow a specific function of time that is shaped by few estimated parameters. For example, if the temporal distribution is assumed to be linear, this condition only requires two parameters, namely an intercept and a slope. This condition is the key feature of MIDAS to establish a balance between model flexibility and parsimony to be able to reasonably interpret the results. This basic model can be enhanced in many different directions, e.g., to whether certain events within the observed period have a different relationship to non-event data [5] or for the evaluation of unequally spaced temporal data [11]. MIDAS has so far been used mainly in accounting and macroeconomics research [e.g., 4, 5, 8]. The method is particularly well suited to accounting research as companies are legally obliged to publish certain economic data such as revenues and costs on a regular low-frequency cycle (e.g., quarterly or annually). Share prices, on the other hand, can generally be retrieved every second and are therefore high-frequency. The job of professional analysts is to

use this information, among other things, to predict the companies' disclosures. For forecasts based on regressions, it is usually necessary to adjust the frequency of the different data samples to the lowest frequency available. For a share price, this could result in the quarterly mean, for example. Therefore, the information within the high-frequency stock market data lost in this process represents a major challenge for analysts to optimize their forecasts. A previous study has shown that MIDAS can significantly help analysts with this challenge and improve the forecasts accordingly [6]. Building on these findings, the next section will discuss whether MIDAS can also help lecturers and researchers in education with specific predictions.

3. MIDAS IN LEARNING ANALYTICS

Prediction models belong to the most used methods in Educational Data Mining [1, 7]. Usually, the corresponding analyses are carried out with classifications or regressions [3]. The prediction of exam grades or scores is one of the most frequently investigated research questions within prediction models [13]. The dependent variables are usually the exam points (regression), the exam grade (classification or regression), or the fact of whether a student has passed or not (classification). In many cases, usage data from learning management systems are used as independent variables. This data is usually high-frequency and must be adjusted to the low-frequency dependent variables for the methods mentioned above. For example, an LMS may record all clicks within the system with precise temporal data. Still, in the above analyses, this information needs to be restricted, for example, to the total number of clicks over the entire period of use [9]. These limitations could be overcome with more sophisticated methods. For instance, it was shown that GARCH, a method from finance research, outperformed the other common methods in multi-modal learning analytics [14]. As of this writing, there is no publication in the field of Educational Data Mining or Learning Analytics that has used the MIDAS approach. This research gap should be filled with the present project. In a subsequent step, MIDAS could even be linked to GARCH to further enrich the research setting [10]. Thus, the research question of this project is whether MIDAS is suitable for predicting exam results in an educational context and how the results compare to those of already known methods in Educational Data Mining. Previous studies have shown that it is important to look not only at aggregate usage data for a given time period but also at the distribution and sequence of the corresponding data points [e.g., 9, 12]. MIDAS could make a valuable contribution to the range of methods already available, as it takes into account the high frequency of independent variables while still providing well-interpretable and thus actionable results for instructors. These results could, for example, be used to build an early warning system for students at risk of academic failure. The good interpretability of MIDAS results could make it easier for instructors to take appropriate measures compared to when using complex machine learning algorithms, whose results might be much more challenging to interpret. Such an early warning system is especially beneficial in lectures where there is little performance feedback between students and teachers in general (e.g., because there is only one final exam at the end of the semester) or due to special conditions (e.g., COVID-19). In such cases, all actors involved see the result of learning and teaching behavior only at the end of the semester through the exam result. Since it is already too late for countermeasures at this point, an early warning system with clear recommendations for action would be beneficial in such a context. Therefore, MIDAS is a promising addition to the current variety of methods and should be considered in future studies.

4. NEXT STEPS

A first application of the basic MIDAS model will be carried out in the following setting as soon as the project's data collection is completed. We developed a mobile learning app for an undergraduate accounting course at a large public university in Europe. The course is compulsory and ought to be taken in the third semester of the bachelor's program. The course is taken by approximately 600 students per semester and consists of a weekly lecture, a biweekly exercise, and biweekly tutorials (five meetings in small groups). The content of this course includes the basics of cost accounting as well as a summary of their significance and classification in the management accounting context. The primary learning material consists of a slide deck, a collection of exercises (with solutions), and a trial exam (all available as PDF files). In the evaluations of earlier semesters, students often complained that there were no contemporary possibilities to learn the subject matter. Therefore, we decided to develop an additional learning tool in the form of a smartphone app, which was launched in the summer semester of 2019. The use of the app is voluntary, and no extra credits or advantages for the final exam can be earned by collecting points in the app. The tool is available via a web version and as an app in the Google Play Store and the Apple App Store. The app's core element is a database with over 550 questions that covers all nine chapters of the course. In addition to the question types single and multiple-choice, there are also sorting and cloze text tasks. The app can be used in three modes: The chapter mode can be used to answer specific questions about a single chapter. As soon as a student has mastered the problems of one chapter, the next chapter is unlocked. In random mode, questions are randomly selected from the chapters that have already been unlocked in chapter mode. In the third mode, the so-called Weekly Challenge, users can compare themselves with other students. Once a week, they have the opportunity to answer 25 questions randomly selected from the chapters already covered in the lecture. The results are subsequently displayed in a weekly and a semester ranking. For good performances in the Weekly Challenge and other learning achievements, students can earn so-called badges, which are then displayed in their account under their self-chosen username. By answering questions (regardless of the mode), students also earn learning points and thus increase their learning level. The progress display of the individual chapters shows students how well they currently master a particular topic. The app has been specifically designed to complement the existing course and is not intended to replace other learning materials such as the slides or the collection of exercises. The app contains an individual explanation for each question which is displayed if a wrong answer is given. Thus, students can work their way through the catalog of questions independently of time and place and eliminate any gaps in their understanding without having to rely on the presence of the lecturers. This is an essential value-added for the students, especially in such a large course with approximately 600 students per semester. The collected app data consists of details about the usage behavior of each student (e.g., time of use, performance (history) regarding every question, and earned badges). At this stage, we already have four semesters of app usage, and the data set is growing as the research project is still ongoing. This is especially promising as the situation regarding COVID-19 lead to an exogenous shock. While in the years 2018 and 2019, the course was held face-to-face, in the summer semester 2020, it was

converted into a purely online lecture. Apart from the launch of the app and the switch to an online lecture, there were no teaching design changes over the course of the semesters. The lecturer and the learning materials remained constant, as well as the design and the grading of the final exam. This unique setting could provide valuable insights into the impact of COVID-19 on higher education. The starting point of the corresponding analysis would be a basic linear regression with the exam score as dependent and usage data from the app as independent variable. The exam score is measured once, while the usage data from the app could be evaluated by every second of the semester. In this setting, we face the challenge of unequal frequencies on both sides of the equation that was described before. If we would only take the sum of total questions answered by a student as the independent variable, we would miss a lot of information. The type and especially the time of usage can be decisive for the effect on the exam score. We would miss all this information with reducing the app usage on measures like total questions answered. Therefore, the MIDAS approach offers a promising possibility to extract more insights from the data set. A comparative analysis with other already known methods in Educational Data Mining or Learning Analytics, which take into account the high frequency of data, could highlight the additional benefits of MIDAS for this research area.

5. CONCLUSION AND FUTURE WORK

In this paper, it was shown that the innovative approach MIDAS could be a promising extension of the variety of methods in Educational Data Mining and Learning Analytics. Further insights will be gained by testing the approach with the usage data from our gamified learning app. Based on the findings, it will be further discussed whether the basic model of MIDAS should be extended for the use in an educational setting or whether other novel methods should be applied in this setting. Besides, it could be promising to apply the MIDAS approach to already published analyses in order to test the corresponding added value. If the results are similarly insightful as those in accounting research, MIDAS could find numerous use cases in Educational Data Mining and Learning Analytics.

6. REFERENCES

[1] Aldowah, H. et al. 2019. Educational data mining and learning analytics for 21st century higher education: A review and synthesis. *Telematics and Informatics*. 37, (2019), 13–49.

[2] Baker, R.S. and Inventado, P.S. 2014. Educational data mining and learning analytics. *Learning analytics*. Springer. 61–75.

[3] Bakhshinategh, B. et al. 2018. Educational data mining applications and tasks: A survey of the last 10 years. *Education and Information Technologies*. 23, 1 (2018), 537–553.

[4] Ball, R.T. et al. 2019. Tilting the evidence: the role of firm-level earnings attributes in the relation between aggregated earnings and gross domestic product. *Review of Accounting Studies*. 24, 2 (2019), 570–592.

[5] Ball, R.T. and Gallo, L.A. 2018. A mixed data sampling approach to accounting research. *Available at SSRN 3250445*. (2018).

[6] Ball, R.T. and Ghysels, E. 2018. Automated earnings forecasts: beat analysts or combine and conquer? *Management Science*. 64, 10 (2018), 4936–4952.

[7] Chen, G. et al. 2020. Let's shine together! a comparative study between learning analytics and educational data mining. *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge* (2020), 544–553.

[8] Clements, M.P. and Galvão, A.B. 2009. Forecasting US output growth using leading indicators: An appraisal using MIDAS models. *Journal of Applied Econometrics*. 24, 7 (2009), 1187–1206.

[9] Conijn, R. et al. 2016. Predicting student performance from LMS data: A comparison of 17 blended courses using Moodle LMS. *IEEE Transactions on Learning Technologies*. 10, 1 (2016), 17–29.

[10] Engle, R.F. et al. 2013. Stock market volatility and macroeconomic fundamentals. *Review of Economics and Statistics*. 95, 3 (2013), 776–797.

[11] Ghysels, E. et al. 2007. MIDAS regressions: Further results and new directions. *Econometric reviews*. 26, 1 (2007), 53–90.

[12] Malekian, D. et al. 2020. Prediction of students' assessment readiness in online learning environments: the sequence matters. *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge* (2020), 382–391.

[13] Romero, C. and Ventura, S. 2010. Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*. 40, 6 (2010), 601–618.

[14] Sharma, K. et al. 2019. Modelling Learners' Behaviour: A Novel Approach Using GARCH with Multimodal Data. *European Conference on Technology Enhanced Learning* (2019), 450–465.

Towards fair, explainable and actionable clustering for learning analytics

Tai Le Quy
Leibniz University Hannover
Hannover, Germany
tai@l3s.de

Eirini Ntoutsi
Freie Universität Berlin
Berlin, Germany
eirini.ntoutsi@fu-berlin.de

ABSTRACT

Clustering is an important technique in learning analytics for partitioning students into groups of similar instances. Application examples include group assignments, students-class allocation, etc. However, traditional clustering does not ensure a fair-representation in terms of some protected attributes like gender or race, and as a result, the resulting clusters might be biased. Moreover, traditional clustering might result in clusters of varying cardinalities reducing their actionability for end user. In many applications, like group assignment, the capacity of the resulting clusters should be controllable to allow direct applicability of the resulting clusters. Furthermore, it is important to be able to explain why an instance/student is clustered into a specific cluster and/or which attributes play a crucial role in the clustering process. We believe that the aforementioned aspects of fairness, capacity and explainability are important for the successful application of clustering in the learning analytics domain.

Keywords

learning analytics, clustering, fairness, bias, explainability, capacity, actionability

1. INTRODUCTION

In education, machine learning (ML) has been used in a wide variety of decision-making tasks, for example, student dropout prediction [11], education admission decisions [25] or forecasting on-time graduation of students [16]. Recently, the incidents of discrimination in ML-based decision-making systems in education, such as grades prediction [4, 15], are an important reason for the increase of the attention to bias and fairness in ML of researchers [32]. Accordingly, the decisions made by the ML-based systems against groups or individuals on the basis of *protected attributes* like gender, race, etc. Bias in education has been studied in many aspects from different sources of bias in education [27], students' data analysis [3], racial bias [39] and gender bias [26].

Tai Le Quy and Eirini Ntoutsi "Towards fair, explainable and actionable clustering for learning analytics". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 847-851. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

However, ML-based decision-making systems have the potential to amplify prevalent biases or create new ones and therefore, fairness-aware ML approaches are required also for the learning environments.

In our research, we are focusing on the *fairness* of clustering methods in learning analytics since clustering is an effective method to analyze student data [8, 17, 28, 36]. Clustering algorithms are useful tools for partitioning students into groups of similar instances [3, 31]. Results from clustering methods are applicable in educational activities such as group assignments [10] and student team achievement divisions [37]. However, the traditional clustering algorithms do not take into account the fairness w.r.t. protected attributes like gender or race, as a consequence of focusing only on the similarity objective. Moreover, the cardinality of the resulting clusters is typically not part of the objective function and as a result clusters of very different cardinalities might be extracted reducing the usefulness of the results. Moreover, understanding the instances-to-clusters assignments, the important features for clustering and what characterizes each cluster (the so-called, cluster labels) is not always easy [33].

The aim of this research is to study the fairness, capacity and explainability requirements and challenges in the learning analytics domain and propose effective solutions that can be used by the domain experts. In this direction, we propose the concept of *fair-capacitated clustering* which extends traditional clustering focusing on clustering quality to also ensure fairness of representation in terms of some protected attribute(s) and the applicability of the resulting clusters by ensuring balanced cluster cardinalities. Such clusters can be exploited by different stakeholders in the learning environment: educators can better organize the learning activities, e.g., group assignments; students can learn better in a more inclusive and equitable environment.

In another direction, we plan to extend the *fair capacitated clustering with explainability* to give insights to the end users about how certain assignment decisions are made, what features are important for clustering and what the extracted clusters represent. Such information will allow educators to customize teaching activities to each group and improve the learning trajectory of each student, each group and the class in overall.

We believe that the results of our research are useful in other domains as well, for example business (clustering cus-

tomers in marketing studies, salesmen areas distribution), traffic (vehicle routing) and communication (network design). Moreover, our research contributes to the further development of the domain of fairness and responsible AI with new methods (for the unsupervised learning problem) and application domain (learning analytics).

The rest of our paper is structured as follows: Section 2 overviews the related work. Research questions are presented in Section 3. Section 4 describes our ongoing work on fair-capacitated clustering and preliminary results. Finally, conclusions and outlook are presented in Section 5.

2. RELATED WORK

Chierichetti et al. [7] first introduced the fair clustering problem and presented a balance measure for computing fairness in the resulting clusters. They defined "fairlet" as a small cluster preserving fairness measure, and then they apply k-Center clustering algorithm on these fairlets to obtain the final clusters. In the later studies, Backurs et al. [1] described an algorithm for the fairlets computation in nearly linear time. The problem of fair clustering with multiple protected attributes is investigated in the researches of Rösner and Schmidt [34] and Bera et al. [2].

The capacitated clustering problem (CCP) was first introduced by Mulvey and Beck [30] with heuristic and subgradient algorithms. Later, researchers proposed approaches to solve the problem in the different clustering methods. For instance, Khuller and Sussmann [19] introduced an approximation algorithm for the capacitated k-Center problem. An improved version of k-Means algorithm for CCP was presented by Geetha et al. [12] with the use of a priority measure to assign points to their centroid. Lam and Mittenthal [20] proposed a heuristic hierarchical clustering method for CCP.

Quite a few researchers, recently, are interested in the usefulness of explainable and interpretable clustering models. Chen et al. [6] proposed a probabilistic discriminative model with the ability to learn rectangular decision rules for each cluster. Saisubramanian et al. [35] offered a voting method to consider which features are meaningful for the end user. Moshkovitz et al. [29] used an unsupervised decision tree to explain k-Means and k-Medians methods.

3. RESEARCH QUESTIONS

We organize the challenges into the research questions $Q1 - Q3$ explained hereafter:

Q_1: **What is fairness in learning analytics and how to mitigate discrimination in clustering?** Fairness in education is an interesting topic researchers [5, 9, 13]. We investigate the fairness terminology in student analytics w.r.t protected attributes such as gender, race. Student performance can be considered as the protected attribute because in some cases no knowledge of the student's performance can help to prevent bias in the grading procedure [23, 24]. Related work in the fairness-aware ML area depicts a large variety of approaches that can be categorized into: i) pre-processing approaches that intervene at the input data [22]; ii) in-processing approaches that directly tweak the clustering algorithm to account for fairness [7] and iii) post-processing approaches that adjust the clustering results to ensure fairness [38]. We will mainly follow the in-processing approaches that directly incorporate fairness in the clustering process. However, such approaches depend on the clustering algorithm per se; our current work focuses on hierarchical and partitioning algorithms, in the future density-based clustering will be also investigated.

Q_2: **How to satisfy multiple objectives, namely capacity of clusters and fairness of representation on top of the (standard) cluster similarity objective?** As already mentioned, the actionability of the results is important. As a concrete example consider group assignments: groups should be comparable to allow for a fair allocation of work among students. In the capacitated clustering problem [30], they do not consider fairness, nor explainability. Likewise approaches for fair clustering also exist [7]. However, approaches that jointly consider the different objectives do not exist.

Q_3: **What is the explanation of a (fair-capacitated) clustering model and how to find it?** The importance of explainable clustering results for the end users has been already discussed. Explainability does not only allow for understanding how certain decisions are made but also allows for debugging of algorithmic decisions and corrections in case of decisions based on protected attributes like gender or race. There are different aspects to explainability in clustering: understanding how a certain assignment of an instance to a cluster was made, understanding what attributes contributed to clustering and explaining what each cluster is about (or cluster labeling). We will investigate the different aspects to allow educators to better understand the groups that are formed and to allow both educators and single users/students to understand how they fit into a particular cluster.

4. PRELIMINARY RESULTS ON FAIR CAPACITATED CLUSTERING

In this section, we present the preliminary results of our work namely *fair-capacitated clustering* [21] problem. The goal is to cluster students into fair-groups w.r.t. single protected attribute. *Gender*, typically, is chosen as the protected attribute. In other words, we would like to balance the number of males and females in the resulting clusters and our proposed methods should satisfy the size of group constraint in order to make the results more actionable.

We define the problem of (t, k, q)-fair-capacitated clustering as finding a clustering $\mathcal{C} = \{C_1, \cdots C_k\}$ that partitions the data X into k clusters such that the cardinality of each cluster $C_i \in \mathcal{C}$ does not exceed a threshold q, i.e., $|C_i| \leq q$ (*the capacity constraint*), the balance of each cluster is at least t, i.e., $balance(\mathcal{C}) \geq t$ (*the fairness constraint*), and minimizes *the objective function*. Parameters k, t, q are user-defined referring to the number of clusters, minimum balance threshold and maximum cluster capacity, respectively.

We present a two-step solution to the problem: i) we rely on fairlets [7] to generate minimal sets that satisfy the fair constraint and ii) we propose two approaches, namely hierarchical clustering (denoted by *hierarchical fair-capacitated*) and partitioning-based clustering (denoted by *k-Medoids fair-capacitated*, to obtain the fair-capacitated clustering. The hierarchical approach embeds the additional cardinality re-

quirements during the merging step while the partitioning-based one alters the assignment step using a knapsack problem formulation to satisfy the additional requirements.

We experiment our proposed methods on four educational datasets: UCI Student performance[1], PISA test scores[2], OULAD[3], MOOC[4], containing the demographics, grades and school-related attributes of students. Table 1 in Appendix A summarizes the characteristics of datasets.

We report on clustering quality (measured as clustering cost, see Eq. 1), cluster fairness (expressed as cluster balance [7], see Eq. 2 and Eq. 3) and cluster capacity (expressed as cluster cardinality). The parameters are set as follows: the minimum threshold of balance $t = 0.5$, i.e., the proportion of the minority group is at least 50% in the resulting clusters; the maximum capacity of clusters $q = \left\lceil \frac{|X| * \epsilon}{k} \right\rceil$; ε is set to 1.01 and 1.2, for k-Medoids fair-capacitated and hierarchical fair-capacitated methods, respectively.

$$\mathcal{L}(X, \mathcal{C}) = \sum_{s_i \in S} \sum_{x \in C_i} d(x, s_i) \tag{1}$$

$$balance(C_i) = \min \left(\frac{|\{x \in C_i | \psi(x) = 0\}|}{|\{x \in C_i | \psi(x) = 1\}|}, \frac{|\{x \in C_i | \psi(x) = 1\}|}{|\{x \in C_i | \psi(x) = 0\}|} \right) \tag{2}$$

$$balance(\mathcal{C}) = \min_{C_i \in C} balance(C_i) \tag{3}$$

The baseline includes well-known clustering methods with fairness-aware approaches and a traditional algorithm. **1) k-Medoids**[18]. This is a traditional partitioning technique of clustering that uses the actual instances as centers (medoids) and divides the dataset into k clusters and minimizes the clustering cost. **2) Vanilla fairlet** [7]. A vanilla fairlet decomposition that ensures fair clusters is generated, then, a k-Center clustering algorithm [14] is applied to cluster those fairlets into k clusters. **3) MCF fairlet** [7]. It is an updated version of the *Vanilla fairlet* with The fairlet decomposition is transformed into a *minimum cost flow* (MCF) problem, by which an optimized version of fairlet decomposition in terms of cost value is computed.

The preliminary results show that our approaches deliver well-balanced clusters in terms of both fairness and cardinality while maintaining a good clustering quality. In terms of clustering cost (Figure 1-a) (Appendix B), our approaches outperform the vanilla fairlet and MCF fairlet methods although they are worse compared to the vanilla k-Medoids clustering. This is obvious due to the fact that our methods have to satisfy constraints on fairness or/and cardinality. *MCF fairlet hierarchical fair-capacitated* shows the best performance due to the optimization in the merging step. As illustrated in Figure 1-b regarding to fairness, our methods are comparative to the competitors. In which, the minimum threshold of balance t is visualized as a dashed line and the actual balance from the dataset is plotted as a dotted line. In Figure 1-c, the maximum capacity thresholds q are presented by the dashed and dotted lines. Our approaches are more preeminent with a lower dispersion, in terms of cardinality. The boxplots of our methods are drawn thicker because the variation of the capacity of resulting clusters is tiny in quite a few cases. MCF fairlet shows the worst performance, followed by Vanilla fairlet and vanilla k-Medoids algorithm.

[1]https://archive.ics.uci.edu/ml/datasets/Student+Performance
[2]https://www.kaggle.com/econdata/pisa-test-scores
[3]https://analyse.kmi.open.ac.uk/open_dataset
[4]https://github.com/kanika-narang/MOOC_Data_Analysis

5. CONCLUSION AND OUTLOOK

The investigations of the fairness, capacity and explainability requirements in the learning analytics domain are the main goals of our research. In this paper, we present the challenges of our work with 3 research questions. The preliminary results on the fair-capacitated clustering problem show that our approaches can satisfy multiple objectives namely fairness, capacity and clustering cost. In the next step, we want to deploy the implementation of an explainable fair clustering algorithm to achieve the clarification of the assignment in a fair clustering method.

Acknowledgements

The work of the first author is supported by the Ministry of Science and Education of Lower Saxony, Germany, within the PhD program "LernMINT: Data-assisted teaching in the MINT subjects", for which the second author is a principal investigator.

6. REFERENCES

[1] A. Backurs, P. Indyk, K. Onak, B. Schieber, A. Vakilian, and T. Wagner. Scalable fair clustering. In *International Conference on Machine Learning*, pages 405–413. PMLR, 2019.

[2] S. Bera, D. Chakrabarty, N. Flores, and M. Negahbani. Fair algorithms for clustering. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NIPS*, volume 32. Curran Associates, Inc., 2019.

[3] S. Bharara, S. Sabitha, and A. Bansal. Application of learning analytics using clustering data mining for students' disposition analysis. *Education and Information Technologies*, 23(2):957–984, 2018.

[4] K. Bhopal and M. Myers. The impact of covid-19 on a level students in england. *SocArXiv*, 2020.

[5] S. Bøyum. Fairness in education–a normative analysis of oecd policy documents. *Journal of Education Policy*, 29(6):856–870, 2014.

[6] J. Chen, Y. Chang, B. Hobbs, P. Castaldi, M. Cho, E. Silverman, and J. Dy. Interpretable clustering via discriminative rectangle mixture model. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 823–828. IEEE, 2016.

[7] F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii. Fair clustering through fairlets. In *NIPS*, pages 5036–5044, 2017.

[8] A. M. De Morais, J. M. Araujo, and E. B. Costa. Monitoring student performance using data clustering and predictive modelling. In *2014 IEEE frontiers in education conference (FIE) proceedings*, pages 1–8. IEEE, 2014.

[9] N. J. Dorans and L. L. Cook. *Fairness in educational assessment and measurement*. Routledge, 2016.

[10] M. Ford and J. Morice. How fair are group assignments? a survey of students and faculty and a modest proposal. *Journal of Information Technology Education: Research*, 2(1):367–378, 2003.

[11] J. Gardner, C. Brooks, and R. Baker. Evaluating the fairness of predictive student models through slicing analysis. In *LAK'19*, pages 225–234, 2019.

[12] S. Geetha, G. Poonthalir, and P. Vanathi. Improved k-means algorithm for capacitated clustering problem. *INFOCOMP*, 8(4):52–59, 2009.

[13] C. Gipps and G. Stobart. Fairness in assessment. In *Educational assessment in the 21st century*, pages 105–118. Springer, 2009.

[14] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

[15] S. Hubble and P. Bolton. A level results in england and the impact on university admissions in 2020-21. *House of Commons Library*, 2020.

[16] S. Hutt, M. Gardner, A. L. Duckworth, and S. K. D'Mello. Evaluating fairness and generalizability in models predicting on-time graduation from college applications. *Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019)*, 2019.

[17] Y. Jayabal and C. Ramanathan. Clustering students based on student's performance-a partial least squares path modeling (pls-pm) study. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 393–407. Springer, 2014.

[18] L. Kaufman and P. J. Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, 344:68–125, 1990.

[19] S. Khuller and Y. J. Sussmann. The capacitated k-center problem. *SIAM Journal on Discrete Mathematics*, 13(3):403–418, 2000.

[20] M. Lam and J. Mittenthal. Capacitated hierarchical clustering heuristic for multi depot location-routing problems. *Int. J. Logist. Res. Appl.*, 16(5):433–444, 2013.

[21] T. Le Quy, A. Roy, G. Friege, and E. Ntoutsi. Fair-capacitated clustering. *The 14th International Conference on Educational Data Mining (EDM 2021)*, 2021.

[22] B. T. Luong, S. Ruggieri, and F. Turini. k-nn as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 502–510, 2011.

[23] J. M. Malouff, A. J. Emmerton, and N. S. Schutte. The risk of a halo bias as a reason to keep students anonymous during grading. *Teaching of Psychology*, 40(3):233–237, 2013.

[24] J. M. Malouff, S. J. Stein, L. N. Bothma, K. Coulter, and A. J. Emmerton. Preventing halo bias in grading the work of university students. *Cogent Psychology*, 1(1):988937, 2014.

[25] F. Marcinkowski, K. Kieslich, C. Starke, and M. Lünich. Implications of ai (un-) fairness in higher education admissions: the effects of perceived ai (un-) fairness on exit, voice and organizational reputation. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 122–130, 2020.

[26] T. Masterson. An empirical analysis of gender bias in education spending in paraguay. *World Development*, 40(3):583–593, 2012.

[27] M. Meaney and T. Fikes. Early-adopter iteration bias and research-praxis bias in the learning analytics ecosystem. In *Companion Proceeding of the 9th International Conference on Learning Analytics & Knowledge (LAK'19), Fairness and Equity in Learning Analytics Systems Workshop*, pages 14–20, 2019.

[28] A. Merceron and K. Yacef. Clustering students to help evaluate learning. In *IFIP World Computer Congress, TC 3*, pages 31–42. Springer, 2004.

[29] M. Moshkovitz, S. Dasgupta, C. Rashtchian, and N. Frost. Explainable k-means and k-medians clustering. In *International Conference on Machine Learning*, pages 7055–7065. PMLR, 2020.

[30] J. M. Mulvey and M. P. Beck. Solving capacitated clustering problems. *European Journal of Operational Research*, 18(3):339–348, 1984.

[31] Á. A. M. Navarro and P. M. Ger. Comparison of clustering algorithms for learning analytics with educational datasets. *IJIMAI*, 5(2):9–16, 2018.

[32] E. Ntoutsi, P. Fafalios, U. Gadiraju, V. Iosifidis, W. Nejdl, M.-E. Vidal, S. Ruggieri, F. Turini, S. Papadopoulos, E. Krasanakis, et al. Bias in data-driven artificial intelligence systems-an introductory survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(3):e1356, 2020.

[33] E. Ntoutsi, M. Spiliopoulou, and Y. Theodoridis. Tracing cluster transitions for different cluster types. *Control & Cybernetics*, 38(1), 2009.

[34] C. Rösner and M. Schmidt. Privacy preserving clustering with constraints. *arXiv preprint arXiv:1802.02497*, 2018.

[35] S. Saisubramanian, S. Galhotra, and S. Zilberstein. Balancing the tradeoff between clustering value and interpretability. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 351–357, 2020.

[36] M. Tanai, J. Kim, and J. H. Chang. Model-based clustering analysis of student data. In *International Conference on Hybrid Information Technology*, pages 669–676. Springer, 2011.

[37] M. Tiantong and S. Teemuangsai. Student team achievement divisions (stad) technique through the moodle to enhance learning achievement. *International Education Studies*, 6(4):85–92, 2013.

[38] C. Vrain, I. Davidson, et al. Constrained clustering via post-processing. In *International Conference on Discovery Science*, pages 53–67. Springer, 2020.

[39] N. Warikoo, S. Sinclair, J. Fei, and D. Jacoby-Senghor. Examining racial bias in education: A new approach. *Educational Researcher*, 45(9):508–514, 2016.

APPENDIX

A. DATASET

Table 1: An overview of the datasets

Dataset	#instances	#attributes	Protected attribute	Balance score
UCI student performance-Mathematics	395	33	Gender (F: 208, M: 187)	0.899
UCI student performance-Portuguese	649	33	Gender (F: 383; M: 266)	0.695
PISA test scores	3,404	24	Male (1: 1,697; 0: 1,707)	0.994
OULAD	4,000	12	Gender (F: 2,000; M: 2,000)	1
MOOC	4,000	21	Gender (F: 2,000; M: 2,000)	1

B. UCI STUDENT PERFORMANCE DATASET

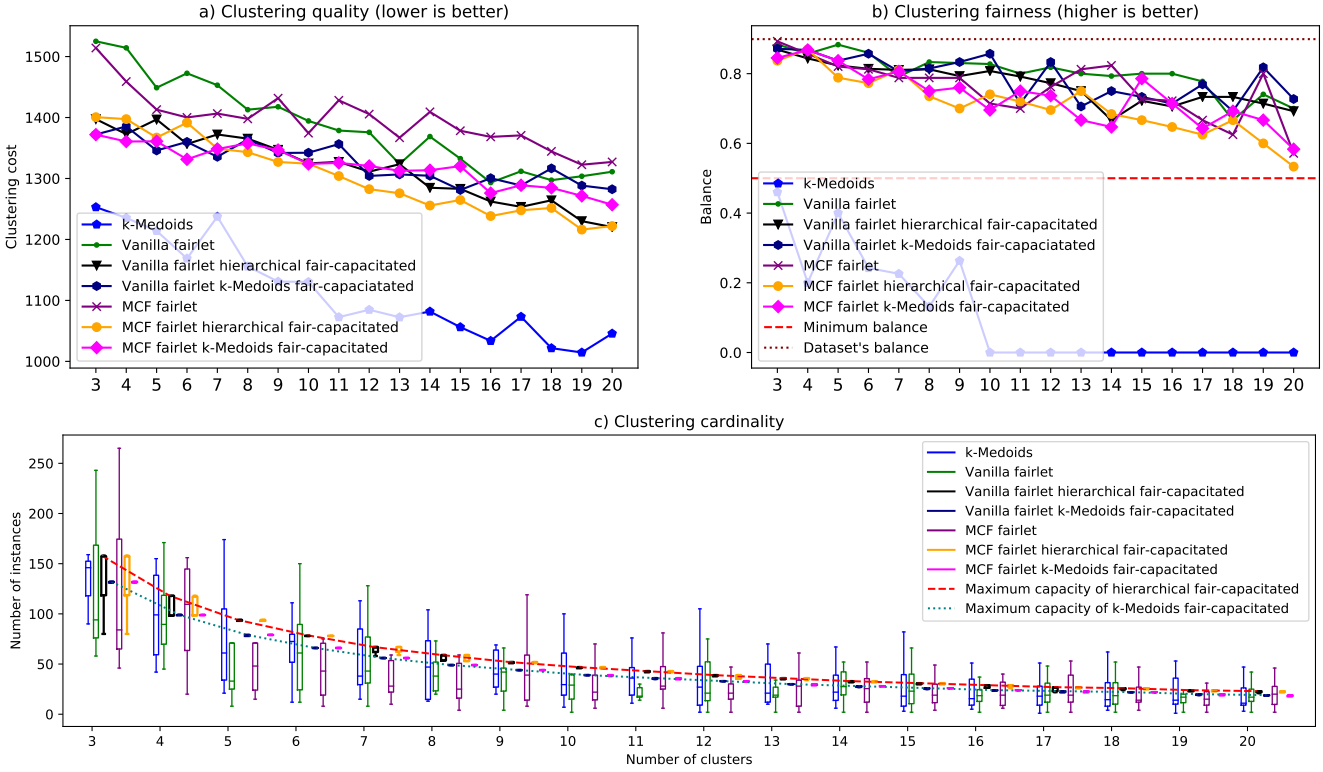


Figure 1: Performance of different methods on UCI student performance dataset - Mathematics subject

TOWARDS A CONCEPTION AND INTEGRATION OF AN EDUCATIONAL SOCIAL NETWORK INTO AN INSTITUTIONAL LEARNING PLATFORM

Romaric BASSOLE[1]
bassoler12@gmail.com,
Frédéric Tounwendyam OUEDRAOGO[1]
frederic.ouedraogo@unz.bf
Laurence Capus[2]
laurence.capus@ift.ulaval.ca
[1]Université Norbert ZONGO,
[2]Université Laval

ABSTRACT

In this article, we take a look at digital social networks in education. The observation made on the campus of Norbert ZONGO University is that the digital device set up on the campus to support the learning and teaching process has not had the support of users who prefer social networks adapted to their smartphone. Most students use digital social networks for exchanges with their peers or teachers, especially with WhatsApp, Facebook about their courses. Yet these technologies are not designed for educational purposes. After a survey of 318 students to take into account the needs of students and teachers, we propose to design an educational social network. This social network will be integrated into a distance learning platform under development as part of a project. We end by presenting the software architecture of our future educational social network.

Keywords

Educational social networks, CEHL, West Africa, University

1. INTRODUCTION

A Computing Environment for Human Learning (CEHL) is a computer environment whose purpose is to lead learners to develop one or more activities favorable to the achievement of educational objectives [1]. They are used to support or encourage learners in learning. The collaborative learning environment is an example of CEHL, designed to promote certain types of interactions including argumentation, explanations, conflict resolution etc. Many more examples of CEHL exist in the scientific literature.

New research work on CEHL has emerged with the new capabilities offered by Internet and new communication and information technologies [2]. The use of social networks in education is a part of this new work. The educational platforms used to support teaching and learning are neglected in the profile of social networks including Facebook, WhatsApp etc. Yet these technologies are not designed for educational purposes.

Romaric Bassole, Frédéric Ouedraogo, Laurence Capus "Towards a Conception and Integration of an Educational Social Network into an Institutional Learning Platform". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 852-855. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

Social networks with an existing educational component often require a monthly or annual subscription and are often not designed in an institutional framework. Students feel the need to use digital platforms in their learning process, which pushes them towards these technologies that are less suited to their context.

In view of this observation, we propose to design an educational social network (ESN) better suited to the West African context. This ESN could be integrated into the new system [3] proposed by a group of researchers with a profile of West African universities.

In the rest of our work, we will present the context of our study then we will present the analysis of the needs carried out for the implementation of a new device and will end with the architecture of our future device.

2. CONTEXT AND STATE OF THE ART

In this section, we present the context of this research project. We also present a small preview on Facebook technology. We end this part by connecting social networks and computing environments for human learning.

2.1 Context

Computing Environments for Human Learning (CEHL) are used to stimulate and support learning among learners. Despite the multitude of learning / teaching platforms that exist, universities in African countries, particularly Norbert ZONGO University (previously University of Koudougou), have difficulty setting up a resource sharing platform better suited to their context. Some environments such as Moodle which is an online learning platform are being implemented in some universities in West Africa to support learning / teaching. This powerful platform is badly used or even abandoned by the first users. Teachers and students are sometimes tempted to use other technology such as WhatsApp [4]–[6] for learning / teaching purposes. To take into account their needs, researchers [3] propose to set up a system better suited to the West African context while keeping services existing.

Students see themselves using other means to share, interact with their peers and teacher. Its means are among others through social

media such as: WhatsApp, Facebook, etc. Social media are used by students as the preferred tools for sharing, communicating, photographing parts of the course / TD. Its tools, easily accessible through their smartphones, are practically used on a daily basis. Although these media have a real advantage, accessibility and flexibility of use [7], it should be noted that they are not suitable for learning / teaching. These technologies are used to create groups without the knowledge of the pedagogy managers which makes it difficult to follow up, see its contribution in improving the learning / teaching of students.

.

To have an CEHL adapted to the West African context, a project [3] is already underway to set up a digital device. Our problem comes in addition to this project but we are focusing more on social networks aspect.

2.2 Curent situation

2.2.1 Social networks

First introduced by Australian anthropologist John A. Barnes [8], a social network is defined as a set of social interactions that unite a group of individuals. These social interactions can be: friendships, family ties, professional ties or specific ties. With the advent of Internet, the notion of social network took a new turn and gave birth to the notion of digital social network. Boyd and Ellison define digital social networks as "a web service allowing individuals to build a profile or not created by a combination of content and, on the other hand, to articulate this public profile with others" [9]. The most famous social networks these days are: Facebook, Twitter, LinkedIn, MySpace etc. these digital networks have drawn the attention of researchers [4], [7], [8] to its possible use in learning / teaching.

2.2.2 Educational social networks

An educational social network is first and foremost a social network. But unlike this one, the individuals in relationships are the learners and the teachers. It is a network that enables teacher-student and student-student, one-to-one, one-to-many and many-to-many interactions. Social networks occupy an important place nowadays in society and its more and more used by young people. This trend has prompted teachers to use these networks in the classroom [4], [6], [9] including Facebook, WhatsApp, etc.

Some educational social networks have been developed to support and stimulate learning among pupils or students. We can cite:

Learndia[1] is an educational social network and interactive learning space dedicated to students. It provides students with course content and a space to simulate assessments. In February 2017, the founders announced the release of the desktop version which does not require an Internet connection.

Freasyway[2] is an international educational social network for students, institutions, independent teachers. Beyond the fact that it offers interactive teaching, this platform offers students the possibility of obtaining information on the types of procedures to be carried out with institutions.

Madabooky [10] is an educational social network targeting terminal and third grade students. Created by three young Madagascans, Tsira Louis Venceslas, Dada Manacé Sylvano and Haritiana Rabemanantsoa, after one of them failed the Baccalauréat exams many times.

These social networks all have in common the objective of stimulating learning among pupils or students. However, these networks raise two major problems:

- Cost: Access to these platforms is conditioned by a subscription to the platform for a flat fee. The cost of these platforms is a barrier for students. In addition to this, the cost of the internet connection is an issue. The questionnaire found that 87% of students use the Internet connection of mobile operators (Orange, Telecel and Moov Africa).
- Institutional scope: These educational social networks have been developed outside the institutions in charge of education (universities, colleges and high schools, training centre, institute, etc.). This causes two major problems, students do not always interact with their teachers and the syllabus may be not consistent with their own courses.

Social networks such as Facebook, WhatsApp are widely used by students nowadays. These easily accessible technologies via smartphones are now used by learners and teachers to learn/teach [5], [6], [9]. These technologies, although used by students for consultation, collaboration, sharing and production activities in their learning processes, were not designed for pedagogical purposes. Moreover, access to user data for integration with existing educational platforms such as Moodle is problematic [6]. Furthermore, these platforms cannot be linked to institutional systems set up to support learning. Finally, the question of accessibility to user data arises with these applications. Yet these data are useful for monitoring learning and for educational research. In an article entitled "How WhatsApp makes Educational Data Mining difficult in West African universities"[6], the authors posed the difficulty for researchers in the field to have educational data for their research.

Educational social networks designed to support learning are inspiring solutions but they do not address the concerns observed on the campus of the Norbert ZONGO University. The design of these educational social networks is not adapted to the system set up in the universities of Burkina Faso. The cost of these applications is a major problem.

In view of this, we proposed an appropriate solution. The following section presents our proposal.

3. PROPOSED SOLUTION

This section of our paper deals with the analysis of students' needs following a survey conducted on a sample of 318 students from public and private universities in Burkina Faso. We propose a new learning device and present its architecture.

3.1 Description of the questionnaire

To understand student practices on campus, we conducted a survey with a student questionnaire. This study concerned 318 students from public and private universities in Burkina Faso.

The questionnaire consists of four parts: student identification, access to communication and information technologies (ICT), use of ICT and use of educational platforms.

[1] https://learndia.com/

[2] https://freasyway.com/public/

The identification of the student made it possible to collect the demographic social information of the students (sex, age group, sector, university, etc.). The section, Access to ICT, allowed us to collect information on the types of electronic support available to students, on Internet access at the University and at home. The last two sections of our questionnaire concerned the use made by students of ICT and educational platforms.

3.2 Needs analysis and specifications

To better understand student practices, we conducted a survey of 318 students from public and private universities in Burkina Faso. This survey allowed us to capture the needs of the students.

Users: Teachers and students are the future users of our solution. Indeed, the network should connect students of the same class and their teachers. Also, students should be able to send invitations to other students (their elders for example). As for teachers, they should be able to send and receive invitations from their colleagues and students. An administrator should ensure the proper functioning of the system.

Analysis of student needs: The survey revealed the following student needs, namely communication, sharing, mutual support:

Communication is the major problem for students. The number of students per teacher is very high so that the students are not satisfied with explanations in class. There is not anyway to contact or ask questions of the teacher outside of the classroom. The students noted that they have class WhatsApp groups to communicate. Although they have this communication tool, their teachers are not included in these groups.

Sharing files, lessons, tutorials, exercise solutions and tutorials are also a major concern for students. The WhatsApp group serves as their tools for sharing but the storage problem is posed. The data of these groups are quickly deleted if the storage disk is full.

Mutual aid appears important when an exercise or another part of a course is misunderstood. Students use Web searches.

As a result of this analysis, our solution should be able to allow students and teachers to make the following actions grouped in this table.

Requirements specification		
Student	Teacher	Admin
▪ Manage an account ▪ Send invitations ▪ Ask questions ▪ Answer questions ▪ Share information	▪ Manage an account ▪ Send invitations ▪ Answer questions ▪ Share information ▪ Create discussion topic	▪ Manage accounts

Tableau 1:requirements specification

3.3 Functional architecture of the new device

Figure 1 shows the software architecture that we have chosen for the development of the future device.

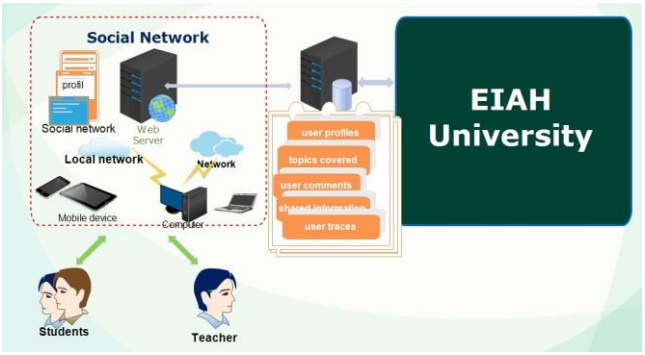


Figure 1: Software architecture

The new device complements existing platforms. This figure below presents an overview of our future system. The users will have access to the social network by Internet or by a local network. To respond to the difficulties related to the accessibility of the Internet connection on campus, students will be able to access the social network through a local network. It will be accessible via smartphones, tablets, laptops and desktops.

This device will be powered by text, audio, video, image and podcast data from students and teachers.

4. Conclusion

In this paper, we have dealt with the establishment of a social network for students from West Africa, particularly those of Burkina Faso. We have presented the background and the objectives of this research. To better understand and take into account the needs of students, we conducted a survey with a questionnaire on 318 students. The results of this questionnaire allowed us to highlight the needs and expectations of students for a better learning environment. We proposed to design an educational social network more suited to the learning context of students in West African universities. We presented the architecture of our future device. Our goal in this research is to create a free and open source platform that will create a community of researchers around this theme.

The work already done and presented in this paper is a first step of this research. The next step will concern the design of this new device. We plan to evaluate our proposed learning system at two levels: pedagogical, technical. To do this, we will enlist experts in education science to our research team.

5. Références

[1] P. Tchounikine, *Précis de recherche en Ingénierie des EIAH*. 2009.

[2] P. Tchounikine et A. Tricot, « Environnements informatiques et apprentissages humains », in *Informatique et sciences cognitives : Influences ou confluence ?*, D. K. Catherine Garbay, Éd. OPHRYS / MSH, 2011, p. 153-186.

[3] M. Bousso AU - L. Capus AU - T.F. Ouédraogo, « A project to create an African e-learning platform more equitable », in

INTED2020 Proceedings, mars 2020, p. 8816-8819, doi: 10.21125/inted.2020.2403.

[4] A. Serge Armel, M. Coulibaly, et T. Karsenti, « WhatsApp: Un enjeu d'enseignement/Apprentissage en Afrique? Enquête auprès des acteurs scolaires au Bénin », *Transmettre*, vol. 1, p. 87-112, déc. 2016.

[5] S. So, « Mobile instant messaging support for teaching and learning in higher education », *Internet High. Educ.*, vol. 31, p. 32-42, oct. 2016, doi: 10.1016/j.iheduc.2016.06.001.

[6] R. BASSOLE, T. F. OUEDRAOGO, et L. CAPUS, « How WhatsApp makes Educational Data Mining difficult in West African universities », *Fairness Account. Transpar. Educ. Data FATED*, 2020.

[7] L. Mélot, A. Strebelle, J. Mahauden, et C. Depover, « Utilisation de Facebook en contexte universitaire », vol. 24, mai 2017, Accessed March 9 2021. http://sticef.org/num/vol2017/24.1.4.melot/24.1.4.melot.htm. sur: http://sticef.org/num/vol2017/24.1.4.melot/24.1.4.melot.htm.

[8] P. Mercklé, « La "découverte" des réseaux sociaux. À propos de John A. Barnes et d'une expérience de traduction collaborative ouverte en sciences sociales », *Réseaux*, vol. 182, nº 6, p. 187-208, 2013, doi: 10.3917/res.182.0187.

[9] Danah m. Boyd et N. B. Ellison, « Social Network Sites: Definition, History, and Scholarship », *J. Comput.-Mediat. Commun.*, vol. 13, nº 1, p. 210-230, oct. 2007, doi: 10.1111/j.1083-6101.2007.00393.x.

[10] Madabooky. https://www.madabooky.com/ (Accessed March , 2021).

Benefits of alternative evaluation methods for Automated Essay Scoring

Øistein E. Andersen
ALTA Institute
University of Cambridge
United Kingdom
oa223@cam.ac.uk

Zheng Yuan
ALTA Institute
University of Cambridge
United Kingdom
zheng.yuan@cl.cam.ac.uk

Rebecca Watson
iLexIR Ltd
Cambridge
United Kingdom
bec@ilexir.co.uk

Kevin Yet Fong Cheung
Cambridge Assessment
University of Cambridge
United Kingdom
Cheung.K@cambridgeenglish.org

ABSTRACT

Automated essay scoring (AES), where natural language processing is applied to score written text, can underpin educational resources in blended and distance learning. AES performance has typically been reported in terms of correlation coefficients or agreement statistics calculated between a system and an expert human examiner. We describe the benefits of alternative methods to evaluate AES systems and, more importantly, facilitate comparison between AES systems and expert human examiners. We employ these methods, together with *multi-marked* test data labelled by 5 expert human examiners, to guide machine learning model development and selection, resulting in models that outperform expert human examiners.

We extend on previous work on a mature feature-based linear ranking perceptron model and also develop a new multi-task learning neural network model built on top of a pre-trained language model – DistilBERT. Combining these two models' scores results in further improvements in performance (compared to that of each single model).

Keywords

Student Assessment, Metrics, Evaluation, Automated Essay Scoring, Natural Language Processing, Deep Learning

1. INTRODUCTION

Automated essay scoring (AES) is the task of employing computer technology to score written text. Learning to write a foreign language well requires a considerable amount of practice and appropriate feedback. On the one hand, AES systems provide a learning environment in which foreign language learners can practice and improve their writing skills even when teachers are not available. On the other hand, AES reduces the workload of examiners and enables large-scale writing assessment. In fact, these technologies have already been deployed in standardised tests such as the TOEFL and GMAT [7, 6] as well as in a classroom setting [26].

As English is one of the world's most widely used languages, and learners naturally outnumber teachers, AES systems aimed at 'English as a Second or Other Language' (ESOL) are in high demand. Consequently, there is a large body of literature with regards to AES systems of text produced by ESOL learners [20, 3, 5, 28, 2, 30, 1, 23, 16], overviews of which can be found in various studies [25, 22, 15].

AES systems exploit textual features in order to measure the overall quality and assign a score to a text. The earliest systems used superficial features, such as essay length, as proxies for understanding the text. As multiple factors influence the quality of texts, later systems have used more sophisticated automated text processing techniques to exploit a large range of textual features that correspond to different properties of text, such as grammar, vocabulary, style, topic relevance, and discourse coherence and cohesion. In addition to lexical and part-of-speech (PoS) n-grams, linguistically deeper features such as types of syntactic constructions, grammatical relations and measures of sentence complexity are some of the properties that form an AES system's internal marking criteria. The final representation of a text typically consists of a vector of features that have been manually selected and tuned to predict a score on a marking scale as accurately as possible, an approach which has involved extensive work on feature development and optimisation.

In contrast, the most recent AES systems are based on neural networks that learn the feature representations automatically, without the need for this kind of manual tuning [1, 23, 19, 16, 27]. Taking the sequence of (one-hot vectors of

Øistein E. Andersen, Rebecca Watson, Zheng Yuan and Kevin Yet Fong Cheung "Benefits of alternative evaluation methods for Automated Essay Scoring". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 856-864. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

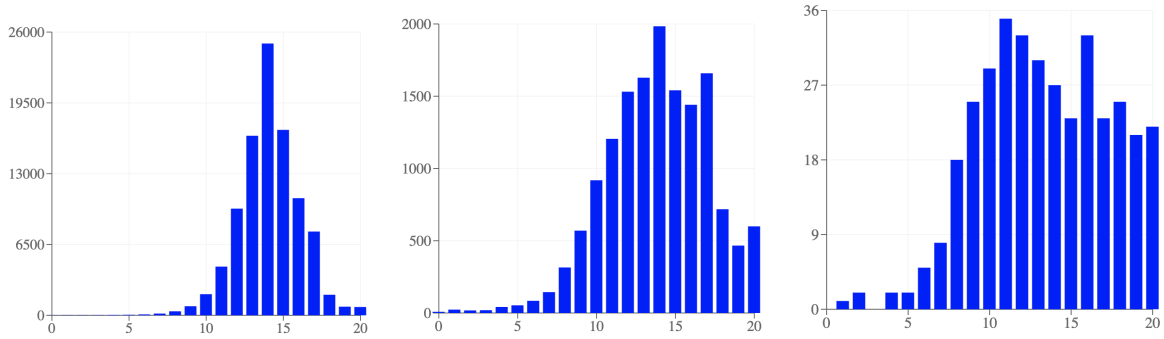


Figure 1: Data distributions (0-20 score on x-axis, count on y-axis). Left to right: Full training set (98,138 responses), u400 training set (14,966), test set (364).

the) words in an essay as input, Alikaniotis et al. [1] and Taghipour et al. [23] studied a number of neural architectures for the AES task and determined that a bidirectional Long Short-Term Memory (LSTM) [14] network was the best performing single architecture. With recent advances in pre-trained bidirectional Transformer [24] language models such as Bidirectional Encoder Representations from Transformers (BERT) [11], pre-trained language models have been applied for AES to achieve state-of-the-art performance [19, 16].

The B2 First exam, formerly known as Cambridge English: First (FCE), is a Cambridge English Qualification that assesses English at an upper-intermediate level. We extend a mature state-of-the-art feature-based AES system [5, 28, 2], researched and developed over the last decade using Cambridge English's FCE exam answers and their corresponding operational scores as training data. Further, we develop a new multi-task learning (MTL) neural network model built on top of a pre-trained masked language model – DistilBERT [21].

Various evaluation metrics have been used to evaluate AES systems, including correlation metrics such as Pearson's Correlation Coefficient (PCC) and Spearman's Correlation Coefficient (SCC), agreement metrics like quadratic weighted Kappa [8] (QWK) and quadratic agreement coefficient [13] (AC2), and error metrics such as Mean Absolute Error (MAE) and Mean Square Error (MSE).

We introduce novel evaluation methods that employ *multi-marked* test data, where each test item has been labelled by more than one expert human examiner, to facilitate comparison of human and AES system performance. Our methods aim to recognise that the set of examiner scores per answer represent an *acceptable range* of scores and thence we aim to evaluate AES systems against this set of scores rather than against a single gold standard score or via inter-rater agreement metrics. This is an important distinction given that expert examiner performance represents the upper bound on the AES task. To the best of our knowledge, this is the first work to perform an in-depth comparison of feature-based and neural-based AES model performance. Further, we illustrate that these models can be considered complementary, and combined to improve performance.

2. DATA

We employ a large training set, collected by Cambridge Assessment,[1] comprising almost 50,000 FCE examination scripts from 2016–20 with operational scores, as well as a newly created multi-marked test set containing 182 scripts labelled by 5 expert human examiners.[2] Each script consists of two questions, and responses are scored using 4 fine-grained assessment scales: content, communicative achievement, organisation and language. Each scale provides a score between 0 and 5 inclusively, and the overall score is calculated by summing over these 4 individual scales to provide an answer score in the range 0–20. For this AES task, we employ the overall 0–20 score to train and test models.[3]

The full training set contains almost 100,000 individual responses to over 50 different prompts, all labelled with a score in the range 0–20, but with an uneven distribution strongly concentrated around 14 (the score expected by an average learner having attained the B2 level for which the exam is designed). In order for the multi-marked test set to include as wide a range of responses as possible, 182 scripts (each consisting of two answers) were sampled to provide a more uniform distribution of scores in the range 16–40 as well as a certain number of lower scores (scripts with scores 0–15 are rarely seen since they correspond to a level far below the one required to pass the exam); the 364 individual answers show a relatively uniform distribution of scores above 8. Similarly, a more balanced training set of just under 15,000 answers was extracted from the full training set by excluding supernumerary scripts from the middle of the scale; u400.[4] The resulting distributions can be seen in Figure 1.

3. METRICS

3.1 Traditional Metrics

Yannakoudakis & Cummins [29] investigated the appropriateness and efficacy of evaluation metrics for AES including

[1] https://www.cambridgeassessment.org.uk/

[2] The operational score, combined with 5 examiner scores, results in 6 scores per answer in the test data. In contrast, the training data contains a single operational score.

[3] Previously, Yannakoudakis et al. [28] worked at the script level (i.e. across two answers) and therefore used scores in the range 0–40.

[4] Note: u400 was selected to be uniformly distributed at the script-level; with 400 randomly selected (maximum) scripts for each script score level 0–40.

SCC, PCC, QWK and AC2 under different experimental conditions. They recommend AC2 [13] for evaluation and reporting SCC and PCC measures for error analysis and system interpretation. Therefore we report these three evaluation metrics (AC2, SCC, PCC), as well as RMSE which we consider operationally desirable; it penalises larger errors more than smaller errors.

Ke & Ng [15] provide a survey of AES system research and popular public corpora employed in evaluation. Most public corpora contain a single human annotator score and evaluation is limited to considering this score the gold standard thence evaluation aids in comparison of AES systems but it is not possible to determine a reasonable upper bound on the task.

The CLC-FCE dataset [28] and the Automated Student Assessment Prize (ASAP) corpus, released as part of a Kaggle competition,[5] include scores assigned by four and two human annotators, respectively. For these, multi-marked corpus evaluation can be performed against a single reference score by taking an average of the scores [1, 16].[6] Alternatively, agreement between the AES system and (each) human expert can be compared to inter-rater agreement performance (which represents the upper bound the task) [28, 19]. Yannakoudakis et al. [28] calculate the average pair-wise agreement across all markers (human examiners and AES system) to produce a single (comparable) metric for SCC and PCC.We perform inter-rater and rater-to-AES pair-wise evaluations for SCC, PCC, AC2 and RMSE in our experimentation, and determine the average performance across the 5 expert human examiners.

3.2 Multi-marked Metrics

We also employ a novel evaluation method whereby scores are only considered to be erroneous if they fall outside the *acceptable range* of scores, as defined by the set of expert human examiner scores considered. We consider two score ranges: i) the range of 5 expert examiner scores (*ALL*) and ii) a narrower range (*MID3*) where we remove the top and bottom scores (for each test item). In addition, we report performance achieved for each of these ranges after removing a single examiner's score from the range, in turn, so that we can compare the performance of each expert examiner against the AES models.

Given a score range, we report the accuracy (percentage of scores that fall within the range) and a novel RMSE variant; RMSE^{R}, which considers the size of the error as equal to the distance between the score and the range. For example, if a score falls above the range we calculate the error as the difference between the score and the highest score in the range.

3.3 RMSE_c Graphs

Operationally, the best performing model may not necessarily be one that achieves the highest performance value based on single metric such as AC2. Rather, a model that performs well *across* the assessment scale is preferable. Further, it is possible for models to achieve similar (single) metric performance but exhibit very different performance distributions across the scale (cf. uniform vs non-uniform distributions with the same average).

Baccianella et al. [4] argued that macro-averaged metrics, including macro-averaged root mean squared error (RMSE^{M}), are more suitable for ordinal regression tasks. RMSE^{M} is calculated by averaging over RMSE_c (RMSE determined for each score c on the assessment scale). That is, RMSE_c is RMSE calculated over the subset of test items that are labelled c. They argue that macro-averaged metrics are more robust to test set distribution given the average results in equally weighting the error rate for each label in the assessment scale. Therefore, we report the RMSE^{M} metric.

We also want to explicitly analyse how a model performs across the assessment scale. Therefore, we employ individual RMSE_c measures, for each reference score c (0–20), and produce novel graphs; RMSE_c graphs, where the score (c) is plotted on the x-axis and the RMSE_c value is plotted on the y-axis. We also produce RMSE^{R}_c graphs, where we calculate RMSE_c values based on our novel RMSE^{R} variant.

4. AES MODELS

4.1 Feature-based

In this work, we extend a mature feature-based AES model [5, 28, 2]: a ranking timed aggregate perceptron (TAP) model trained on a set of features shown to encode the information required to distinguish between texts exhibiting different levels of language proficiency attained by upper-intermedite learners. Features include ones that can be extracted directly from the text (word and character n-grams) or a parsed representation (PoS n-grams and parse rule names), as well as various statistics (PoS categories, lengths, readability scores, use of cohesive devices, etc.) and error estimations (rule-based and corpus-based). We also include features that measure congruence between question and answer (similarity between embeddings for different parts), but that is not the focus of this paper.

Unlike for models used in previous work, the n-gram features have been filtered to exclude ones that encode punctuation without context; this forces the model to focus on other, possibly more relevant, aspects of the text and at the same time removes the possibility of artificially inflating model scores by adding superfluous punctuation characters. The models trained on the full and u400 training sets will be referred to as the TAP and TAP_1, respectively, in the following.

4.2 Neural Network

In recent years, fine-tuning pre-trained masked language models like BERT via supervised learning has become the key to achieving state-of-the-art performance in various natural language processing (NLP) tasks. These models often consist of over 100 million parameters across multiple layers and have been pre-trained on large amounts of existing text data to capture context-sensitive meaning of, and relations between, words. Following [19, 16], our neural approach builds upon this, where we use pre-trained DistilBERT as

[5] https://www.kaggle.com/c/asap-aes

[6] For ASAP, the *resolved* score is often employed, which is calculated as the average between the two human examiner scores (if the scores are close), or is determined by a third examiner (if the scores are far apart).

Table 1: Average inter-rater and rater-to-AES performance (Ex1–Ex5)

	Op	Ex1	Ex2	Ex3	Ex4	Ex5	TAP	TAP_1	NN	TAP+NN	TAP_1+NN
SCC	0.74	0.77	0.72	0.75	0.74	0.77	0.75	0.74	0.78	**0.79**	0.78
PCC	0.73	0.76	0.69	0.76	0.75	0.76	0.74	0.73	**0.78**	**0.78**	0.77
AC2	0.90	0.92	0.92	**0.94**	**0.94**	**0.94**	**0.94**	0.93	**0.94**	**0.94**	**0.94**
RMSE	2.74	2.41	2.44	2.19	2.19	2.25	2.20	2.21	2.09	2.08	**2.05**

Table 2: RMSE using average examiner (Ex1–Ex5) scores (ExAvg).

	TAP	TAP_1	NN	TAP+NN	TAP_1+NN
RMSE	1.70	1.72	1.58	1.56	**1.52**
$RMSE^M$	1.70	1.34	1.55	1.54	**1.33**

the basis for our neural network model and add additional layers on top to perform supervised tasks. We choose DistilBERT for practical reasons – it retains 97% of the language understanding capabilities of BERT, while reducing parameter size by 40% and decreasing model inference time by 60% [21].

We treat AES as a *sequence regression* problem and construct the input by adding a special start token ([CLS]) to the full text:

$$[\text{CLS}], w_1, w_2, \ldots, w_t, \ldots, w_n \tag{1}$$

This representation is then used as input to the output layer to perform regression.

Compared with feature-based models, for neural network models to be effective, they need to be trained on a large amount of annotated data. MTL allows models to learn from multiple objectives via shared representations, using information from related tasks to boost performance on tasks for which there is limited target data [18, 10, 31, 9]. Instead of only predicting the score of an essay, we extended the model to incorporate auxiliary objectives. The information from these auxiliary objectives is propagated into the weights of the model during training, without requiring the extra labels at testing time. Inspired by the linguistic features used in the feature-based AES systems, we experimented with a number of linguistic auxiliary tasks, and identified the dependency parsing as the most effective one.

The neural AES model is developed as a MTL neural network model trained jointly to perform AES and Grammatical Relation (GR) prediction. Model weights are shared among these two training objectives. The final layer for the AES objective is a fully connected layer that performs regression (i.e. scoring head), while another linear layer is introduced to perform token-level classification to predict the type of the GR in which the current token is a dependent (i.e. classification head). The overall loss function is a weighted sum of the essay scoring loss (measured as MSE) and the dependency parsing loss (as cross-entropy):

$$\text{Loss} = \lambda\,\text{Loss}_{\text{AES}} + (1-\lambda)\text{Loss}_{\text{GR}} \tag{2}$$

During training the whole model is optimised in an end-to-end manner. We refer to the neural MTL model trained on the full training set as the *NN* model in Section 5.

Table 3: Accuracy for ALL range.

		-Ex1	-Ex2	-Ex3	-Ex4	-Ex5
Op	61.3	54.1	55.5	56.0	56.0	59.3
Ex1	*	73.4	*	*	*	*
Ex2	*	*	69.0	*	*	*
Ex3	*	*	*	76.4	*	*
Ex4	*	*	*	*	73.6	*
Ex5	*	*	*	*	*	80.8
TAP	82.1	76.1	76.4	79.4	76.9	79.7
TAP_1	78.8	71.4	72.8	73.9	74.7	76.1
NN	81.0	75.0	76.9	76.1	76.4	78.0
TAP+NN	84.9	**78.8**	79.1	79.9	**81.0**	**82.4**
TAP_1+NN	**85.4**	77.5	**80.8**	**80.5**	80.5	82.1

Table 4: Accuracy for MID3 range.

		-Ex1	-Ex2	-Ex3	-Ex4	-Ex5
Op	36.0	25.5	27.2	26.4	28.3	26.9
Ex1	*	46.2	*	*	*	*
Ex2	*	*	43.1	*	*	*
Ex3	*	*	*	42.9	*	*
Ex4	*	*	*	*	40.9	*
Ex5	*	*	*	*	*	**50.0**
TAP	59.9	46.4	**49.5**	45.1	**49.2**	46.7
TAP_1	53.6	44.5	45.6	41.5	41.5	42.6
NN	58.2	43.4	45.9	42.6	43.7	43.7
TAP+NN	**61.8**	47.0	49.2	**46.7**	47.3	48.1
TAP_1+NN	59.6	**47.3**	46.7	45.1	44.0	45.9

5. EVALUATION

To facilitate comparison between AES systems and human examiners, we employed traditional evaluation metrics as described in §3.1. Table 1 shows average inter-rater or rater-to-AES performance in terms of SCC, PCC, AC2 and RMSE calculated between 1) operational scores (Op), scores assigned by an expert (Ex1–Ex5) or scores predicted by an AES system, and 2) each of the experts' scores (excluding the expert being evaluated, if any).[7] For instance:

$$\text{SCC(Ex3)} = \frac{1}{n-1}\sum_{i\neq 3}\text{SCC(Ex3, Ex}i\text{)} \tag{3}$$

For each metric (row) in Table 1, we have highlighted the best performance in bold. AC2 scores 7 of the 10 models the same (top) score of 0.94 and thence, in our experimentation, does not aid in system comparison. Apart from AC2, these traditional evaluation metrics indicate that the NN model outperforms all examiners and feature-based (TAP) models. Both TAP models perform comparatively to the individual examiners, that is, fall in the performance range achieved by examiners (Ex1–Ex5). Performance of the combined TAP and NN models (the average score) is shown in the last two columns of Table 1. Based on these traditional

[7]For interested readers, we have included pair-wise results for SCC, PCC, AC2 and RMSE metrics in the Appendix.

Table 5: $RMSE^R$ for ALL range.

		-Ex1	-Ex2	-Ex3	-Ex4	-Ex5
Op	1.35	1.48	1.46	1.49	1.46	1.43
Ex1	*	1.12	*	*	*	*
Ex2	*	*	1.16	*	*	*
Ex3	*	*	*	0.77	*	*
Ex4	*	*	*	*	0.78	*
Ex5	*	*	*	*	*	0.93
TAP	0.74	0.90	0.92	0.82	0.84	0.79
TAP_1	0.71	0.87	0.85	0.83	0.83	0.81
NN	0.64	0.81	0.74	0.76	0.77	0.70
TAP+NN	0.62	0.79	0.76	0.73	0.74	0.68
TAP_1+NN	**0.58**	**0.74**	**0.68**	**0.68**	**0.68**	**0.65**

Table 6: $RMSE^R$ for MID3 range.

		-Ex1	-Ex2	-Ex3	-Ex4	-Ex5
Op	1.84	2.11	2.03	2.12	2.04	2.04
Ex1	*	1.77	*	*	*	*
Ex2	*	*	1.77	*	*	*
Ex3	*	*	*	1.42	*	*
Ex4	*	*	*	*	1.41	*
Ex5	*	*	*	*	*	1.48
TAP	1.21	1.41	1.49	1.42	1.55	1.42
TAP_1	1.21	1.51	1.44	1.43	1.52	1.46
NN	1.09	1.38	1.31	1.33	1.40	1.31
TAP+NN	1.08	1.32	1.32	1.30	1.41	1.28
TAP_1+NN	**1.01**	**1.31**	**1.23**	**1.25**	**1.34**	**1.25**

metrics, it is unclear whether combining models improves performance. PCC and AC2 indicate no improvement is made over the single NN model, while SCC and RMSE indicate that TAP+NN and TAP_1+NN are best, respectively.

Table 2 compares the AES systems using RMSE and $RMSE^M$ calculated using the average examiner scores (ExAvg) as the single reference score. The combined TAP_1+NN achieves the best RMSE and $RMSE^M$ performance (in line with average examiner RMSE performance in Table 1). $RMSE^M$ is the only metric that illustrates a large performance difference between TAP and TAP_1 models. In fact, TAP_1 significantly outperforms the NN model as well for this metric, indicating that this model performs better across the assessment scale than the other AES models. RMSE and $RMSE^M$, over ExAvg scores, suggest that there is some small performance gains made by combining models.

In addition to traditional evaluation methods, we employed novel multi-marked metrics, as described in §3.2. Tables 3 and 4 illustrate the accuracy (percentage of scores that fall in range) over the ALL and MID3 ranges, respectively. Tables 5 and 6 show the corresponding $RMSE^R$ performance for these ranges, respectively. For all four tables, performance is directly comparable within each column, with the highest accuracy highlighted in bold.[8] The most important evaluation relates to the first column for the ALL range in Tables 3 and 5, as these results compare the performance of the AES models evaluated against all 5 examiner scores' range. Other columns in these tables (-ExN) facilitate comparison between the AES systems and each human examiner (N).

[8]Note, the asterisk symbol in these four tables indicate that the score is part of the acceptable range.

Accuracy and $RMSE^R$ metrics are complementary, as accuracy represents the proportion of scores that are correct while $RMSE^R$ evaluates the degree to which scores fall outside the range of human examiner scores. Operationally, we consider $RMSE^R$ more important than accuracy, given AES systems should be consistent and errors, when they do occur, should be penalised to a greater degree as the scores falls further outside the range of human examiner scores.

Tables 5 and 6 suggest that NN outperforms both TAP models and all human examiners, while both TAP models perform comparatively to the individual examiners; in line with evaluation based on traditional metrics in Table 1. However, in contrast to the metrics discussed thus far, the $RMSE^R$ metric indicates combined models outperform their corresponding individual models. This improvement is more evident for TAP_1+NN, which outperforms all human examiners and AES models across both ranges.

As described in §3.3, we produced novel $RMSE_c$ graphs to compare model performance across the assessment scale. $RMSE_c$ (and $RMSE_c^R$) graphs for the single and combined AES models are shown in Figure 2. The Op and ExAvg graphs plot $RMSE_c$ calculated against the operational and average examiner scores (i.e. c on the x-axis), respectively. The bottom graph, a $RMSE_c^R$ graph, plots the $RMSE^R$ performance for the ALL range where the c score (x-axis) is the average examiner score in the ALL range (i.e. using the same distribution of test items as the ExAvg $RMSE_c$ graph).

Comparing the AES models across the assessment scale, we can see that all AES models follow a similar pattern; they perform better in the mid ranges and worse in the lower and upper score ranges. This finding is not unexpected, given we have ample training data in the mid ranges and very little training data in the upper and lower ranges of the assessment scale (see Figure 1). The TAP_1 model, trained over a more uniformly distributed training set trades smaller declines in performance in the middle of the scale for more consistent results across the scale, in line with the $RMSE^M$ evaluation metric. The NN model achieves better performance in the upper and lower scores compared to TAP, suggesting that it is more robust over skewed training datasets. However, as evident in these $RMSE_c$ graphs, the TAP and NN models tend to perform better in particular ranges of the scale and thence these models are complementary, and combined models benefit from the relative strengths of individual models across the scale.

6. CONCLUSIONS

We deployed two types of AES systems: feature-based and neural network. We found that the NN model is more robust over skewed datasets as it achieves better performance in the upper and lower scores. However, the feature-based models are more interpretable, require significantly less computational overhead to train and can be trained over much smaller datasets than neural-based models. The TAP_1 model, trained over a more uniform subset of the training data performed more consistently than NN across the assessment scale. We illustrated that feature-based TAP and NN models are complementary, and combined models benefit from the relative strengths of individual models across the scale, outperforming human examiners. In operational deploy-

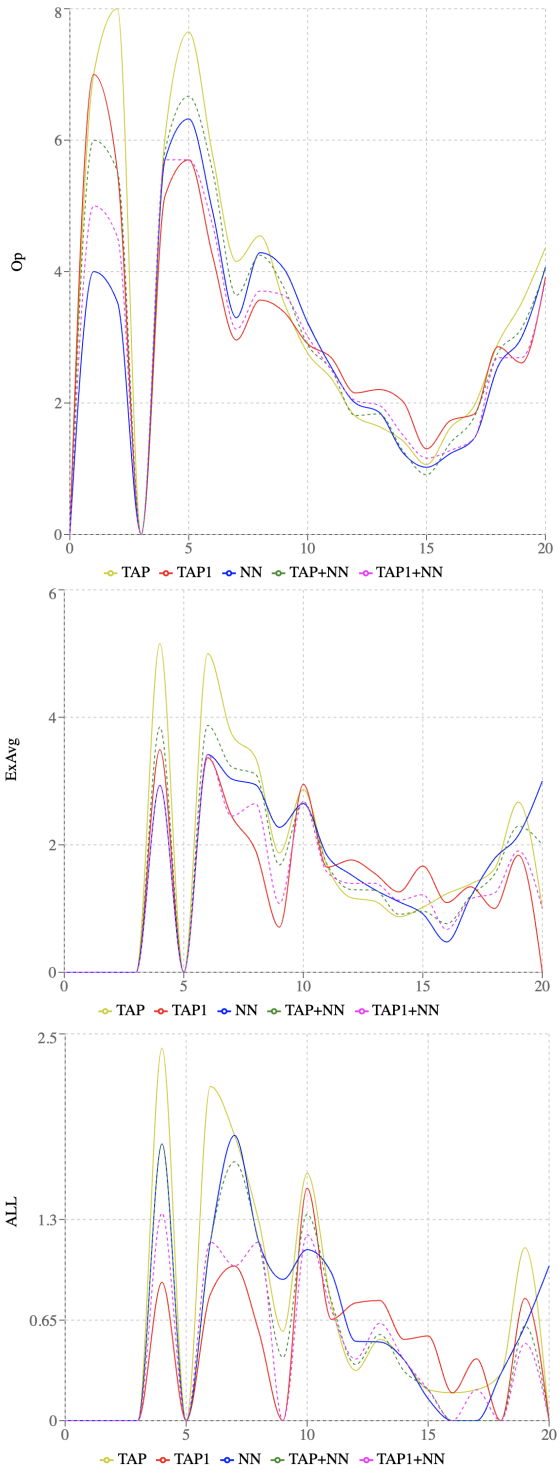


Figure 2: $RMSE_c$ graphs for operational score (Op) and average examiner score (ExAvg). $RMSE_c^R$ graph for the ALL range.

ment, the best performing TAP_1+NN model can make effective use of the constantly growing set of training data by retraining TAP_1 frequently to incorporate any new information available and only retraining the NN models over the full training set from time to time.

We presented novel approaches to evaluating AES that make use of multi-marked/annotated data. These approaches have advantages over traditional evaluation methods and also demonstrate the value of using resources to repeatedly annotate essays for the AES context. Building on the recommendations made by Yannakoudakis & Cummins [29], we make the following observations and suggestions for those working on AES:

- In addition to $RMSE^M$, we recommend calculating $RMSE_c$ and plotting $RMSE_c$ graphs to explicitly analyse how system performance varies across an assessment scale.
- We recommend that, where feasible, a proportion of texts in evaluation sets should be annotated by multiple examiners to allow different forms of evaluation that account for rating variability exhibited by human examiners.
- Where multiple human-derived scores are available, system performance should be evaluated using methods that incorporate the range of scores given for each text. We recommend using a novel RMSE variant; $RMSE^R$, that considers the size of the error as equal to the distance between the score and the upper or lower bound of the range.
- Where multiple human-derived scores are available, we also recommend that the accuracy of a system is calculated, by treating texts scored within the range of scores provided by humans as correct classifications.

Further work is needed to explore the evaluation approaches proposed here to establish how they vary in different contexts, to inform how they should be interpreted. For example, we expect these evaluation metrics to behave differently according to the granularity of the reporting scale, the distribution of evaluation sets and the inter-rater reliability observed between human examiners. Therefore, work to systematically investigate these measures in terms of their robustness to trait prevalence, robustness to marginal homogeneity and robustness to scale scores should be conducted systematically, in a similar vein to simulations reported by Yannakoudakis & Cummins [29].

We have demonstrated the value of producing multi-marked data to support evaluation. However, our proposed metrics can be refined further to allow for more sophisticated uses of multi-marked data, by incorporating methods commonly used for psychometric evaluation and quality assurance, such as Many-Facet Rasch Measurement [17, 12]. Further work should explore how these methods can account for examiner reliability issues when making use of multi-marked data.

7. ACKNOWLEDGMENTS

We would like to thank Ted Briscoe, Michael Corrigan, Helen Yannakoudakis and the anonymous reviewers for their

valuable comments and suggestions. This paper reports on research supported by Cambridge Assessment, University of Cambridge.

8. REFERENCES

[1] D. Alikaniotis, H. Yannakoudakis, and M. Rei. Automatic text scoring using neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 715–725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[2] Ø. E. Andersen, H. Yannakoudakis, F. Barker, and T. Parish. Developing and testing a self-assessment and tutoring system. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 32–41, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[3] Y. Attali and J. Burstein. Automated essay scoring with e-rater® v.2. *The Journal of Technology, Learning and Assessment*, 4(3), Feb. 2006.

[4] S. Baccianella, A. Esuli, and F. Sebastiani. Evaluation measures for ordinal regression. pages 283–287, 01 2009.

[5] T. Briscoe, B. Medlock, and Ø. Andersen. Automated assessment of ESOL free text examinations. Technical Report UCAM-CL-TR-790, University of Cambridge, Computer Laboratory, Nov. 2010.

[6] J. Chen, J. H. Fife, I. I. Bejar, and A. A. Rupp. Building e-rater® Scoring Models Using Machine Learning Methods. *ETS Research Report Series*, 2016(1):1–12, June 2016.

[7] M. Chodorow and J. Burstein. Beyond essay length: Evaluating e-rater®'s performance on toefl® essays. *ETS Research Report Series*, 2004(1):i–38, 2004.

[8] J. Cohen. Inter-rater reliability: Dependency on trait prevalence and marginal homogeneity. *Psychological bulletin*, 4(70):213–220, 1968.

[9] H. Craighead, A. Caines, P. Buttery, and H. Yannakoudakis. Investigating the effect of auxiliary objectives for the automated grading of learner English speech transcriptions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2258–2269, Online, July 2020. Association for Computational Linguistics.

[10] R. Cummins and M. Rei. Neural multi-task learning in automated assessment. *CoRR*, abs/1801.06830, 2018.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[12] S. Goodwin. A many-facet rasch analysis comparing essay rater behavior on an academic english reading/writing test used for two purposes. *Assessing Writing*, 30:21–31, 2016. Innovation in rubric use: Exploring different dimensions.

[13] K. Gwet. Inter-rater reliability: Dependency on trait prevalence and marginal homogeneity. *Stat Methods Inter-Rater Reliab Assess*, 2, 01 2002.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] Z. Ke and V. Ng. Automated essay scoring: A survey of the state of the art. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 6300–6308. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[16] E. Mayfield and A. W. Black. Should you fine-tune BERT for automated essay scoring? In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 151–162, Seattle, WA, USA â†' Online, July 2020. Association for Computational Linguistics.

[17] C. Myford and E. Wolfe. Detecting and measuring rater effects using many-facet rasch measurement: Part ii. *Journal of applied measurement*, 5:189–227, 02 2004.

[18] M. Rei and H. Yannakoudakis. Auxiliary objectives for neural error detection models. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 33–43, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[19] P. U. Rodriguez, A. Jafari, and C. M. Ormerod. Language models and automated essay scoring. *CoRR*, abs/1909.09482, 2019.

[20] L. M. Rudner and T. Liang. Automated essay scoring using bayes' theorem. *The Journal of Technology, Learning and Assessment*, 1(2), June 2002.

[21] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing*, Vancouver BC, Canada, Dec. 2020.

[22] M. D. Shermis and J. Burstein, editors. *Handbook of Automated Essay Evaluation*. Routledge, 2013.

[23] K. Taghipour and H. T. Ng. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1891, Austin, Texas, Nov. 2016. Association for Computational Linguistics.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[25] D. M. Williamson, X. Xi, and F. J. Breyer. A framework for evaluation and use of automated scoring. *Educational Measurement: Issues and Practice*, 31(1):2–13, 2012.

[26] J. Wilson, D. Chen, M. P. Sandbank, and M. Hebert. Generalizability of automated scores of writing quality in grades 3-5. *Journal of Educational Psychology*, 111(4):619–640, May 2019.

[27] R. Yang, J. Cao, Z. Wen, Y. Wu, and X. He.

Enhancing automated essay scoring performance via fine-tuning pre-trained language models with combination of regression and ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1560–1569, Online, Nov. 2020. Association for Computational Linguistics.

[28] H. Yannakoudakis, T. Briscoe, and B. Medlock. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[29] H. Yannakoudakis and R. Cummins. Evaluating the performance of automated text scoring systems. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 213–223, Denver, Colorado, June 2015. Association for Computational Linguistics.

[30] H. Yannakoudakis, Øistein E Andersen, A. Geranpayeh, T. Briscoe, and D. Nicholls. Developing an automated writing placement system for esl learners. *Applied Measurement in Education*, 31(3):251–267, 2018.

[31] Z. Yuan, F. Stahlberg, M. Rei, B. Byrne, and H. Yannakoudakis. Neural and FST-based approaches to grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 228–239, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

APPENDIX

A. FULL PAIR-WISE RESULTS

We include, in the Appendix, individual pair-wise inter-rater and rater-to-AES performance, across the 5 examiners, for operational scores (Op), each human examiner (Ex1–Ex5) and the AES models for SCC, PCC, AC2 and RMSE. Results in the last row in each table, the average of the Ex1–Ex5 scores in each column, can be seen in Table 1 .

Table 7: SCC (best score per row shown in bold).

	Op	Ex1	Ex2	Ex3	Ex4	Ex5	TAP	TAP_1	NN	TAP+NN	TAP_1+NN
Op	∗	0.76	0.69	0.76	0.72	0.75	0.73	0.73	**0.79**	0.77	0.77
Ex1	0.76	∗	0.69	0.79	0.76	0.82	0.80	0.80	**0.84**	**0.84**	**0.84**
Ex2	0.69	0.69	∗	0.73	**0.74**	0.72	0.69	0.66	0.72	0.72	0.70
Ex3	0.76	0.79	0.73	∗	0.73	0.77	0.75	0.74	**0.80**	**0.80**	0.78
Ex4	0.72	**0.76**	0.74	0.73	∗	0.75	0.72	0.73	0.75	**0.76**	**0.76**
Ex5	0.75	**0.82**	0.72	0.77	0.75	∗	0.78	0.77	**0.82**	0.81	0.81
Avg (Ex1–Ex5)	0.74	0.77	0.72	0.75	0.74	0.77	0.75	0.74	0.78	**0.79**	0.78

Table 8: PCC (best score per row shown in bold).

	Op	Ex1	Ex2	Ex3	Ex4	Ex5	TAP	TAP_1	NN	TAP+NN	TAP_1+NN
Op	∗	0.75	0.68	0.76	0.73	0.72	0.73	0.74	**0.77**	**0.77**	**0.77**
Ex1	0.75	∗	0.66	0.79	0.76	0.82	0.79	0.79	**0.83**	**0.83**	**0.83**
Ex2	0.68	0.66	∗	**0.71**	0.70	0.68	0.68	0.65	0.69	0.70	0.69
Ex3	0.76	0.79	0.71	∗	0.76	0.79	0.75	0.73	**0.80**	**0.80**	0.79
Ex4	0.73	0.76	0.70	0.76	∗	0.77	0.73	0.74	0.76	**0.77**	**0.77**
Ex5	0.72	**0.82**	0.68	0.79	0.77	∗	0.76	0.76	0.81	0.81	0.80
Avg (Ex1–Ex5)	0.73	0.76	0.69	0.76	0.75	0.76	0.74	0.73	**0.78**	**0.78**	0.77

Table 9: AC2 (best score per row shown in bold).

	Op	Ex1	Ex2	Ex3	Ex4	Ex5	TAP	TAP_1	NN	TAP+NN	TAP_1+NN
Op	∗	0.90	0.88	**0.91**	0.89	0.90	0.88	0.89	0.89	0.89	0.90
Ex1	0.90	∗	0.90	0.93	0.93	0.94	0.93	0.94	0.94	0.94	**0.95**
Ex2	0.88	0.90	∗	**0.94**	0.92	0.93	0.92	0.90	0.92	0.92	0.92
Ex3	0.91	0.93	0.94	∗	**0.95**	**0.95**	0.94	0.94	**0.95**	**0.95**	**0.95**
Ex4	0.89	0.93	0.92	**0.95**	∗	**0.95**	0.94	0.93	0.94	0.94	0.94
Ex5	0.90	0.94	0.93	**0.95**	**0.95**	∗	0.94	0.94	**0.95**	**0.95**	**0.95**
Avg (Ex1–Ex5)	0.90	0.92	0.92	**0.94**	**0.94**	**0.94**	**0.94**	0.93	**0.94**	**0.94**	**0.94**

Table 10: RMSE (best score per row shown in bold).

	Op	Ex1	Ex2	Ex3	Ex4	Ex5	TAP	TAP_1	NN	TAP+NN	TAP_1+NN
Op	∗	2.72	2.92	2.58	2.72	2.78	2.93	2.71	2.74	2.79	**2.64**
Ex1	2.72	∗	2.77	2.30	2.30	2.28	2.29	2.15	2.05	2.10	**1.99**
Ex2	2.92	2.77	∗	2.30	2.20	2.48	2.22	2.40	2.26	**2.17**	2.24
Ex3	2.58	2.30	2.30	∗	2.08	2.07	2.20	2.24	2.06	2.07	**2.05**
Ex4	2.72	2.30	2.20	2.08	∗	2.15	1.95	2.01	1.90	1.85	**1.84**
Ex5	2.78	2.28	2.48	2.07	2.15	∗	2.34	2.25	2.20	2.21	**2.13**
Avg (Ex1–Ex5)	2.74	2.41	2.44	2.19	2.19	2.25	2.20	2.21	2.09	2.08	**2.05**

Methods for Language Learning Assessment at Scale: Duolingo Case Study

Lucy Portnoff
Duolingo
lucy@duolingo.com

Erin Gustafson
Duolingo
erin@duolingo.com

Joseph Rollinson
Duolingo
joseph@duolingo.com

Klinton Bicknell
Duolingo
klinton@duolingo.com

ABSTRACT

Students using self-directed learning platforms, such as Duolingo, cannot be adequately assessed relying solely on responses to standard learning exercises due to a lack of control over learners' choices in how to utilize the platform: for example, how learners choose to sequence their studying and how much they choose to revisit old material. To provide accurate and well-controlled measurement of learner achievement, Duolingo developed two methods for injecting test items into the platform, which combined with Educational Data Mining techniques yield insights important for product development and curriculum design. We briefly discuss the unique characteristics and advantages of these two systems - Checkpoint Quiz and Review Exercises. We then present a case study investigating how different study approaches on Duolingo relate to learning outcomes as measured by these assessments. We demonstrate some of the unique benefits of these systems and show how educational data mining approaches are central to making use of this assessment data.

Keywords

online learning; language learning; assessment; regression

1. INTRODUCTION

Online learning platforms have at their disposal large volumes of data about how students engage with learning material, how they navigate educational software, and how the learning process unfolds over time. Using a variety of methods - machine learning, statistics, psychometrics, etc. - Educational Data Mining (EDM) and Learning Analytics (LA) researchers identify students at risk of dropout from a course [e.g., 13], detect changes in study behavior [e.g., 11], predict exam performance [e.g., 1, 4, 12], and characterize the different learning strategies that learners adopt [e.g., 1, 12].

Duolingo is a learning platform that provides free language education through mobile apps and a website. With around 40 million users active on the platform each month, Duolingo may well possess the largest language learning dataset of any company or research institution. Researchers at Duolingo leverage EDM/LA methodologies to mine datasets - including internal assessment and log data - for insights that inform improvements to the learning experience, help identify opportunities for changes to curriculum design, and fuel research on second language (L2) learning more generally.

Due to the self-directed nature of the Duolingo learning platform and the desire for holistic learner assessment, we have developed two assessment systems - the Checkpoint Quiz and Review Exercises - that allow for carefully controlled measurement of learner achievement. These two assessments were designed with the challenges gamified platforms struggle with in mind, including ensuring the learning experience remains motivating and maintaining a scalable content creation process.

The utility of the Checkpoint Quiz and Review Exercises for assessing learner achievement depends, at least in part, on the high volume of data collected from Duolingo learners and the EDM methodologies that can be applied to that data. By leveraging predictive modeling and natural language processing (NLP) methods, we are able to control for the various ways that learners choose to navigate through the platform. Further, these methods allow us to uncover useful insights into how this variation in user navigation relates to learning outcomes - insights that we can leverage for product development and curriculum design. In this paper, we present two of our assessment systems and a case study highlighting the importance of applying EDM methodologies to derive insights from Duolingo assessment and log data.

2. RELATED WORK

Most EDM/LA applications at Duolingo focus on pedagogy-oriented issues [10] or computer-supported predictive analytics [2]. Most relevant to the current work are studies focused on predicting performance on upcoming course exercises [9] and predicting performance on an assessment [1, 4, 12]

Rather than relying on assessment data, some systems discussed in other studies instead model student interaction with and performance on individual course exercises. Knowledge tracing [7] is a popular approach for maintaining a model of whether students have learned specific concepts in a course. One system [9] compared the performance of a Bayesian Knowledge Tracing (BKT) model with a Deep Knowledge Tracing (DKT) model using Long Short-Term Memory (LSTM) to better capture longer-

Lucy Portnoff, Erin Gustafson, Joseph Rollinson and Klinton Bicknell "Methods for Language Learning Assessment at Scale: Duolingo Case Study". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 865-871. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

term learning. These models predicted future performance on exercise x_{t+1} given the previous performance record for a student (x_0, ..., x_t). This system treats every student interaction as an opportunity for assessment and the model output was used for developing student-facing modules for progress tracking and content recommendation. However, knowledge tracing approaches primarily focus on characterizing mastery of specific concepts rather than providing a holistic assessment of knowledge or achievement.

Other studies rely both on knowledge tracing and assessment data to analyze course effectiveness and provide this more holistic view. This approach is especially useful in more self-directed learning platforms. One study [4] used BKT to characterize learning using a digital game and used outputs from these models to predict post-test scores following a period of learning with the game. They found that mastery scores for two knowledge components (output from BKT models) had positive and significant association with post-test scores. Insights from the BKT model itself were also useful for identifying concepts that are difficult for students to master, which highlights opportunities for improving course effectiveness. This study also found evidence that learners have poor meta-cognition about their mastery of key concepts; when left to use the learning platform freely, many students continue to practice concepts the BKT model predicts they have mastered rather than moving on to new material.

Knowledge tracing is not the only approach used for characterizing student behavior using clickstream or log data. To make log data useful for predictive modeling, many researchers turn to methods from NLP to aggregate events [1, 12]. Simple methods include calculating n-grams for particular event types. For example, unigrams can capture the number of times a student completes a particular learning module and bigrams can capture the number of times students complete two modules in sequence [12]. Such data can be used as inputs into predictive models either relying solely on raw n-gram counts [12] or by processing the data further using unsupervised machine learning methods - such as hierarchical clustering - to identify common sequence patterns [1].

3. DUOLINGO ASSESSMENT SYSTEMS

3.1 Duolingo Course Structure

Duolingo courses are organized into a series of *units*, each of which concludes with a *Checkpoint*. Courses used by the majority of learners have the following structure: 25-30 *skills* per unit with five difficulty *levels* per skill and 5-6 *lessons* per level. Skills are designed around a particular theme (e.g., Travel). The vocabulary taught in the skill is aligned around that theme (e.g., hotel, airport, passport) and grammatical topics tend to be consistent across lessons within a skill. Lessons typically consist of 12-15 exercises designed to teach some vocabulary and/or grammatical concept. Duolingo curriculum designers incorporate aspects of spiral curriculum [5] to revisit familiar concepts in more complex contexts in future skills. See Figure 1 for an example of the typical Duolingo course structure.

The five levels for each skill provide a scaffolded learning experience, where learners review the same vocabulary or grammatical concepts in increasingly difficult contexts. All skills start with a foundational Level 0 and as learners "level up" a skill they see the same sequence of lessons teaching the same content but using different exercise types. Early levels include exercises that focus on passive recognition, such as matching a second language (L2) word/picture pair with the corresponding word in the first language (L1); see Figure 2). Exercises in later levels are more difficult, as they require recall and production in the L2 (e.g., translating an L1 sentence into L2; see Figure 2). The level achieved for a given skill is indicated in the user interface with a number inside a crown icon (see Figure 1).

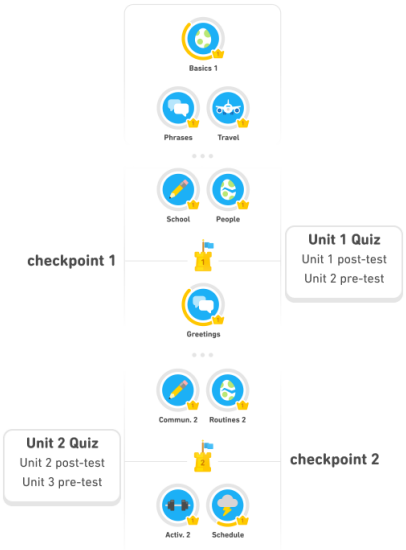


Figure 1. Duolingo course and Checkpoint Quiz design.

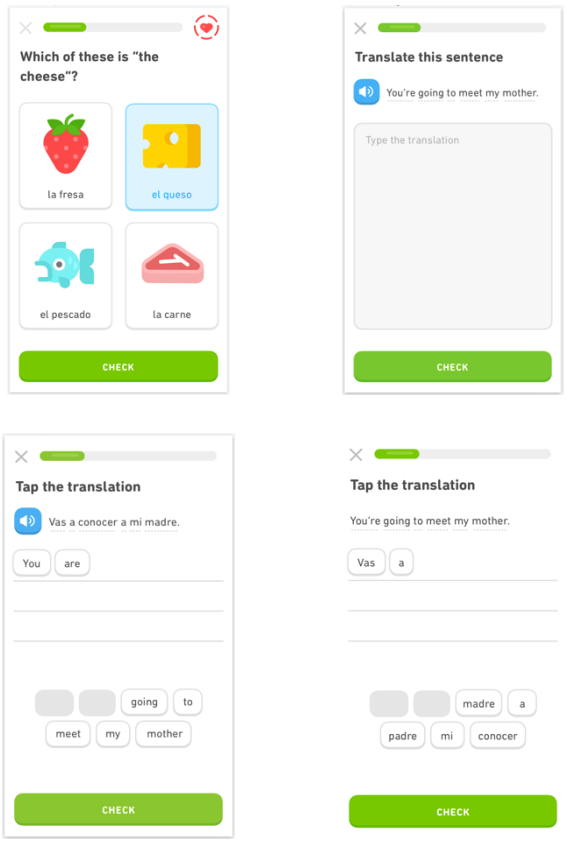


Figure 2. Example exercise types. Top left: passive recognition; top right: recall and production. Bottom left: recall L2→L1; bottom right: recall L1→L2.

When learners begin a Duolingo course, not all skills in the first unit are immediately available; a row unlocks once Level 0 is complete for all skills in the prior row. For example, only the Basics 1 skill is available at first and the next set of skills in the row below Basics 1 (e.g., Phrases and Travel; see Figure 1) will only unlock once Basics 1 reaches Level 1. Once skills are unlocked, learners are free to return to them to practice previously studied material and "level up" the skill. Duolingo learners are, therefore, given agency to choose their learning path. Some learners prefer to attempt only the foundational level in a skill (Level 0) before moving on to new material, while others prefer to level up all skills. Leveling up is entirely optional and learners are required to complete only the foundational level for each skill before they can move on to the next unit of content. This self-directed nature of the learning platform provides challenges for assessing learner achievement.

Other modes of learning are available to users outside of course skills. Learners can build reading and listening proficiency through the Stories feature, which reinforces unit content through interactive dialogues with exercises to check comprehension. Learners can also complete generalized practice sessions, which drill users on content they have already studied from throughout the course. Further, after learners have leveled a skill up all the way, they can return for skill practice to reinforce their knowledge. If learners find skill material too easy, they also have the option to "test out" of a level and jump to harder exercises at the next level.

We use a variety of methods to assess learner achievement and proficiency throughout a Duolingo course. In the sections below, we describe two of the core assessments in use today: Checkpoint Quiz and Review Exercises.

3.2 Checkpoint Quiz

For a subset of Duolingo's courses, learners must complete a custom-built assessment once they finish a unit and reach a Checkpoint. The Checkpoint Quiz is an achievement test that measures the extent to which our learners have achieved the objectives for each unit of a course. Checkpoint Quiz items are independent from the items used in course skills and users are only exposed to the quiz items during the assessment. This ensures that learners do not have the opportunity to learn the items in the assessment while studying course content and is important for test validity. Checkpoint Quiz items were designed by curriculum experts and Duolingo assessment scientists have conducted analyses to ensure their quality.

Learners do not receive corrective feedback or a final grade for the assessment and may only take the quiz once. At each Checkpoint, learners complete a randomly generated quiz consisting of 15 items (sampled from a larger pool of items). Seven items are pre-test items that test the next unit of the course that the learner is about to start and another seven are post-test items that test the unit the learner just completed (critically, the same seven items the learner saw in the previous quiz as a pre-test). This pre-test / post-test design allows us to establish a baseline level of performance so we can later assess gain in accuracy from pre-test to post-test. The final item is a self-directed writing item designed to assess the current unit (with no pre-test). See Figure 1 for an illustration of Checkpoint Quiz design.

The assessment tests knowledge of vocabulary, grammar, listening comprehension, reading comprehension, and free-form writing using separate items designed to test one of these language skills and components. Vocabulary and grammar items are a combination of multiple choice and fill-in-the-blank questions (i.e., learners type the missing word), listening and reading are exclusively multiple choice, and writing questions are free-response. Each item is accompanied by a set of curated tags for grammatical concepts and communicative components.

3.3 Review Exercises

Review Exercises prompt learners to review content from a skill earlier in their course. A single Review Exercise is inserted into randomly selected lessons in the foundational level of a skill (only for skills beyond the first five in the course). These exercises are randomly and uniformly sampled from the pool of available exercises from either three skills or five skills earlier in the course. For example, randomly selected exercises from the Animals skill are injected into Level 0 lessons seen by learners studying the Places skill (see Figure 3). These exercises are inserted into the lesson in a random position, as long as it is not among the first two or last two exercises. Therefore, lessons with Review Exercises will be one exercise longer than a standard lesson. Review Exercises come in two forms: assisted recall and translation from L1-to-L2 or vice versa (see bottom row of Figure 2).

Figure 3. Review Exercise design for testing five skills earlier in course.

Review Exercises as a form of assessment have a number of advantages over the Checkpoint Quiz: 1) Review Exercises are available in all courses; 2) they allow us to measure learning at every skill in a course, rather than just at unit-terminal Checkpoints; and 3) they provide an order of magnitude more data than Checkpoint Quizzes.

However, Review Exercises have a few disadvantages over the Checkpoint Quiz. One key disadvantage is that the items used for Review Exercises overlap with items used for lessons with skills; therefore, we sacrifice some test validity in order to be able to use the assessment at scale across all courses and all skills in a given course. Further, the sentences used as Review Challenges have not been assessed for their quality as measures of learning. Another disadvantage is that the data is not tagged for grammatical concepts or communicative components, which

limits the insights this assessment can provide for informing curriculum design.

Table 1. Key differences between the Checkpoint Quiz and Review Exercises.

Checkpoint Quiz	Review Exercises
Slow data collection (only at Checkpoints)	Fast data collection (at every skill in a course)
Tagged and calibrated by curriculum experts	Not tagged or calibrated
Items siloed from course	Items sampled from course
Only certain courses	All courses

4. CASE STUDY

Learners on Duolingo use the platform in a variety of different ways. In this case study, we investigate how learning decisions impact outcomes, so that we could "nudge" learners to use the app more effectively.

This case study demonstrates how EDM methodologies allow us to investigate the various ways that learners choose to navigate through the platform - focusing on differences in "leveling up" behavior - and how course navigation relates to learning outcomes. We show a correlation between leveling up and higher accuracy on the Checkpoint Quiz. Complementary modeling with Review Exercise data establishes a causal link between completing sessions in higher levels and accuracy on assessments.

4.1 Checkpoint Quiz

4.1.1 Data

Our work uses four months of Checkpoint Quiz data. For every learner completing at least two consecutive Checkpoint Quizzes within this timeframe, we collected the pre-test / post-test item response pairs (e.g., the pre-test responses collected at Checkpoint 1 and the corresponding post-test responses collected at Checkpoint 2) as well as summary statistics on learners' studying behavior in the unit the items assess (e.g., number of lessons completed at each level across skills in Unit 2, number of Stories completed between pre-test and post-test). Responses to free-form writing items were not included in this analysis.

4.1.2 Methods

To isolate the impact of lessons completed at each level on Checkpoint Quiz outcomes, we built a logistic regression model to predict post-test scores for items that were answered incorrectly in the pre-test (a measure of learning gain). Primary variables of interest capture the number of lessons learners completed at a given level for each skill in the unit of interest (frequency counts for Level 1 through Level 4; e.g., a learner completed 20 Level 1 lessons, 15 Level 2 lessons, etc.). Although Duolingo has five levels for all skills (starting with Level 0), we exclude counts for the foundational level because all learners must complete the same number of Level 0 lessons to finish a unit. The model controls for item and user covariates: language component of the item (e.g., vocabulary), unit (e.g., Unit 2), course (e.g., French for English Speakers), number of sessions completed for other types of study material (e.g., Stories, generalized practice, test-outs), self-reported prior proficiency (0-10), and subscriber status[1] (non-paying or paying learner).

4.1.3 Results

We found that average post-test item accuracy increases linearly with every skill-level completed (Figure 4). In other words, each additional level completed across all skills increases the odds of answering a Checkpoint Quiz item correctly by the end of the unit.

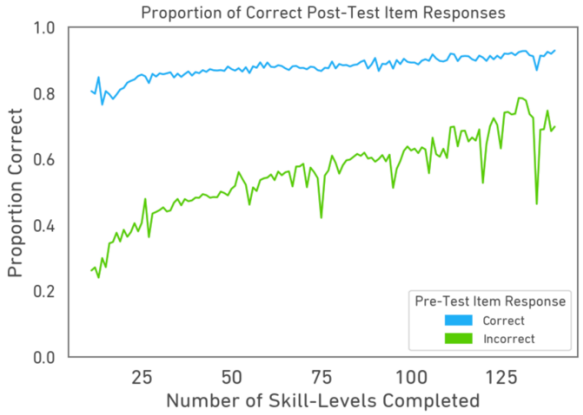


Figure 4. Average post-test accuracy by the number of skill-levels completed as a function of pre-test accuracy.

This finding was supported by the results of our logistic regression model (summarized in Figure 5). We observed that the probability of answering a post-test item correctly increases with every additional lesson in Levels 1, 2, and 4. Level 3 has a negative coefficient, but this is likely an artifact of variable suppression[2].

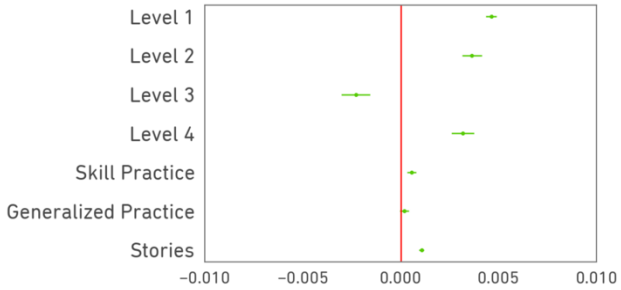


Figure 5. Checkpoint Quiz logistic regression model output. Coefficients of the number of times a user completed seven different session types in a model including other user and item covariates (see Section 4.1.2).

[1] Duolingo offers a paid subscription that removes ads, allows offline access, and includes additional features and learning modes. All learners have access to the same course content.

[2] Because learners tend to complete the same number of lessons in Levels 3 and 4, we attributed the negative coefficient to the statistical consequence of highly collinear relationships existing in the correlation matrix, which can cause variable suppression and model instability [8]. To verify that this multicollinearity did not result in model instability, we repeatedly fit the model on bootstrapped samples of the original data. We found that small changes to the data do not cause any erratic changes in the coefficients, so we concluded that our model estimates are stable.

We also compared the magnitudes of the leveling up effects with those of other types of learning modes, specifically Stories (interactive dialogues to practice reading and listening skills), skill practice, and generalized practice (see Section 2 for more details about these learning modes). Coefficients capturing leveling up behavior show dominant effects in the model; one additional skill-level has a greater impact on Checkpoint Quiz scores than one additional Story, skill practice, or generalized practice.

The Checkpoint Quiz findings show that providing learners with multiple difficulty levels to practice study material improves learning outcomes. Further, we found evidence that completing lessons at Levels 1, 2, 4 is not only positively associated with learning outcomes, but is *more* positively associated than any other activity. However, the Checkpoint Quiz analysis is not necessarily causal. The findings could also be due to self-selection biases, wherein the type of learner that is motivated to complete additional (non-required) levels is likely to perform better in general. A complementary analysis is required to establish a causal link.

4.2 Review Exercises

We utilized Review Exercise data to establish a causal link between leveling up and better learning outcomes. Review Exercises are better suited to this complementary analysis than the Checkpoint Quiz because each Review Exercise targets material from a single source lesson. This design allows us to compare learners who exhibit the same studying behavior except for the completion of one additional level for that lesson. Isolating the change in accuracy from one additional level means that we have controlled for self-selection biases and can interpret the change as causal.

4.2.1 Data

For the Review Exercise analysis, we collected all Review Exercises completed over the course of approximately two months. Data comes from all Duolingo courses. Along with Review Exercise response accuracy, we collect important control variables: whether the exercise came from 3 or 5 skills earlier in the course, exercise type, and the skill the exercise was sampled from (see Figure 3 for Review Challenge design).

4.2.2 Methods

Using logistic regression and a regression discontinuity design (RDD) [3, 6], we are able to model the impact of completing higher levels on Review Exercise accuracy while controlling for self-selection bias that may occur for learners who choose to level up vs. those who do not. An RDD is a quasi-experimental approach where a synthetic treatment condition is assigned to observations that fall above or below a certain "cut-off" point. We achieve this by first identifying learners who have completed any lessons at a given level for the skill a Review Exercise was sampled from (e.g., learners who have completed at least one Level 1 lesson). Among those learners, we define a cut-off point to compare those who have completed that level for the Review Exercise source lesson (e.g., Level 1) to those who have completed that level for the lesson that *immediately precedes* the source lesson but who have not yet completed that level for the source lesson itself (e.g., preceding lesson to Level 1, but source lesson to Level 0). This approach controls for most potential self-selection bias in deciding to level up (all comparisons include learners who have chosen to level up the skill) and can provide stronger evidence for a causal relationship between leveling up and Review Exercise accuracy.

We created a variable with eight levels for use in the regression model to capture 1) the highest level a learner has leveled up the Review Exercise source lesson to and 2) whether the learner studied the source lesson to the same level as the preceding lesson (e.g., both at Level 1) or studied the source lesson one time less than the preceding lesson (e.g., preceding lesson at Level 2 but source lesson at Level 1). For example, this scheme yields coefficients of the form `Level 1:Same Level`, indicating learners for whom both the source lesson and preceding lesson were at Level 1, or `Level 1:Lower Level`, indicating learners for whom the source lesson was at Level 1 and the preceding lesson was at Level 2. This coding scheme required excluding certain observations. Cases where the learner had completed the highest level possible for the Review Exercise source lesson (i.e., Level 4) is not included because it is impossible for the lesson preceding the source lesson to be leveled up any higher. We also exclude observations where the source lesson is the first lesson of a skill because there will be no preceding lesson to serve as a control comparison.

In addition to this main variable, we also control for other factors that influence Review Exercise accuracy: the number of skills away from the source skill (three or five), and the exercise type of the Review Exercise (L1-to-L2 translation or vice versa), and the difficulty of the source skill. We defined difficulty of source skills by computing the log-odds of answering a Review Exercise correctly in each skill in the data overall[3]. This allows us to control for the fact that, all else being equal, accuracy is likely to be lower overall for Review Exercises sampled from more difficult skills, which increases the power of the analysis.

4.2.3 Results

If leveling up causes higher Review Exercise accuracy, we expected to see that the `Level N:Same Level` (source lesson and preceding lesson to Level N) coefficients were significantly larger than the `Level N-1:Lower Level` (source lesson one level lower than preceding lesson; Levels N-1 and N, respectively) coefficients. Such an effect would indicate that - controlling for leveling up behavior overall - completing higher levels of the lesson a Review Exercise came from yields significant improvements in Review Exercise accuracy.

Figure 6 summarizes the results of our logistic regression model. We can see that `Level 1:Same Level` is significantly higher than `Level 0:Lower Level`. This effect indicates that learners who have studied a Review Exercise source lesson twice (at Level 0 and Level 1) are more likely to provide a correct response on their Review Exercise than learners who have studied a Review Exercise source lesson once (only at Level 0) but already had studied the previous lesson twice (at Level 0 and Level 1). This result provides evidence for a causal relationship between leveling up study material and assessment performance, at least for the first time learners level up. The model shows similar trends for leveling up beyond Level 1 (e.g., `Level 2:Same Level` is numerically higher than `Level 1:Lower Level`), suggesting this relationship continues to exist as users study the Review Exercise source lesson additional times (although perhaps with diminishing returns).

The regression results also show significant differences between `Level 0:Same Level` / `Level 0:Lower Level` and

[3] Empirical log odds defined as *log((correct + 1) / (incorrect + 1))*.

`Level 1:Same Level` / `Level 1:Lower Level`. Although the learners captured in the `Lower Level` coefficients had not leveled up the source lesson to Level 1, we see clear improvements in Review Exercise accuracy stemming from leveling up any lessons preceding the source lesson. These learners will not have had additional opportunity to study the exact exercise used for the Review Exercise, but the content and concepts in other lessons in the skill will have been related. Therefore, the benefit of studying in one lesson transfers to other lessons.

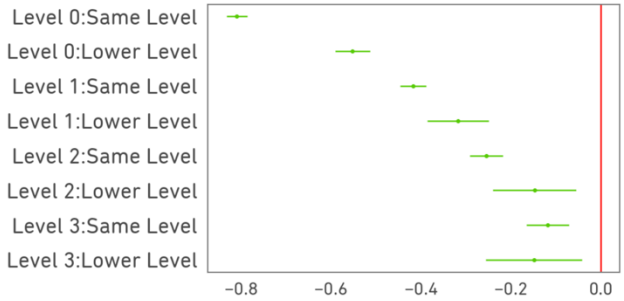


Figure 6. Review Exercises model output. Coefficients of leveling up behavior in a model including other item covariates (see Section 4.2.2).

5. CONCLUSIONS

In a case study of the levels mechanic, wherein learners study content in increasingly difficult contexts by "leveling up", complementary analyses of the Checkpoint Quiz and Review Exercises showed that completing sessions in higher levels leads to stronger performance on assessments. Analyzing accuracy rates on the Checkpoint Quiz by the number of skill-levels completed in the course unit revealed a strong positive trend. Because variation in how learners navigate the platform may introduce self-selection bias and complicate interpretation of these results, we conducted an additional analysis of Review Exercises that controlled for this bias. The Review Exercises analysis supports a causal link between leveling up and improved assessment performance, showing that completing additional levels for a skill (beyond the foundational level) has measurable learning value.

Together, these results directly motivated the implementation of a number of interventions that encourage learners to reach higher levels. For example, because learner awareness of the existence and purpose of levels was relatively low, we added design elements that give learners a visual stand-in for how the levels system works. Learners also now receive a pop-up with a redirect button upon finishing a level prompting them to start the next level in the skill. Randomized controlled experiments (i.e., A/B tests) introducing these changes showed >10% increases in the number of lessons completed in each level beyond the required foundational level and significantly more studying activity on the app overall. These interventions exemplify how insights from the Checkpoint Quiz and Review Exercises have lasting impact on the Duolingo learning experience.

This study focused on one type of variation in how learners choose to navigate the Duolingo learning platform, namely leveling up. Learners can additionally choose their own study sequence for the skills (e.g., completing all the levels in a skill before starting the next skill, completing the entire course unit one level at a time, leveling up clusters of skills within a unit), as well as which types of learning material to study (e.g., course skills, generalized practice, Stories). Future iterations of this work will aim to capture such variation, thereby improving model fit and deepening our understanding of how other types of navigational choices relate to learning outcomes. Previous EDM studies [1, 9] provide methodologies that can be used to characterize this variation.

Future work will also continue to explore the utility and limitations of the Review Exercise assessment system. For example, data from Review Exercises show promise as a method for measuring learning improvements over the course of an A/B test due to the high volume of daily data generated, highly localized measurement (i.e., testing learning of content from specific course skills), and the distributed nature of the assessment (i.e., testing learning in all course skills). Future work could also consider whether Review Exercise accuracy can be predicted based on engagement with (and accuracy on) source lessons in the past.

Self-directed learning platforms such as Duolingo require accurate and well-controlled assessments to measure learner achievement. Because learners exercise a high degree of agency in how they navigate the courses, achievement cannot be adequately assessed by analyzing exercise responses alone. Duolingo developed two forms of assessment - the Checkpoint Quiz and Review Exercises - to capture insights about how different study approaches relate to learning outcomes. Applying EDM techniques to these assessments yields useful insights that inform our understanding of how the navigation of course content relates to learning outcomes and how we can leverage these insights to improve the learning experience on the platform.

6. ACKNOWLEDGMENTS

Special thanks to Daniel Falabella, Xiangying Jiang, Geoff LaFlair, Bozena Pajak, and Karin Tsai for helpful comments on this work.

7. REFERENCES

[1] Nil-Jana Akpinar, Aaditya Ramdas and Umit Acar. 2020. Analyzing Student Strategies in Blended Courses Using Clickstream Data. In *Proceedings of the 13th International Conference on Educational Data Mining (EDM 2020),* July 10-13, 2020, 6-17.

[2] Hanan Aldowah, Hosam Al-Samarraie, & Wan Mohamad Fauzy. 2019. Educational data mining and learning analytics for 21st century higher education: A review and synthesis. *Telemat Inform*, 37 (Apr. 2018), 13–49. https://doi.org/10.1016/j.tele.2019.01.007

[3] Joshua D. Angrist & Jörn-Steffen Pischke. 2014. *Mastering Metrics: The Path from Cause to Effect*. 2014. Princeton University Press, Princeton, NJ.

[4] Huy Anh Nguyen, Xinying Hou, John Stamper, & Bruce M McLaren. 2020. Moving beyond Test Scores: Analyzing the Effectiveness of a Digital Learning Game through Learning Analytics. In *Proceedings of the 13th International Conference on Educational Data Mining* (*EDM 2020)*, July 10-13, 2020. 487–495.

[5] Jerome S. Bruner. 1960. *The Process of Education*. Harvard University Press, Cambridge, MA.

[6] Thomas D. Cook, Donald T. Campbell, & William Shadish. 2002. *Experimental and quasi-experimental designs for generalized causal inference.* Houghton Mifflin, Boston, MA.

[7] Albert T. Corbett & John R. Anderson. 1995. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Model User-Adapted Interaction* 4, 4 (March 1995), 253–278.

[8] Lynn Friedman & Melanie Wall. 2005. Graphical Views of Suppression and Multicollinearity in Multiple Linear Regression. *Am Stat* 59, 2, 127-136. https://doi.org/10.1198/000313005X41337

[9] Tao Huang, Zhi Li, Hao Zhang, Huali Yang, & Hekun Xie. EAnalyst : Toward Understanding Large-scale Educational Data. In *Proceedings of the 13th International Conference on Educational Data Mining (EDM 2020)*, July 10-13, 2020, 620–623.

[10] Zacharoula Papamitsiou, & Anastasios A. Economides. 2014. Learning analytics and educational data mining in practice: A systematic literature review of empirical evidence. *Educational Technology and Society* 17, 4, 49–64.

[11] Jihyun Park, Kameryn Denaro, Fernando Rodriguez, Padhraic Smyth, & Mark Warschauer. 2017. Detecting Changes in Student Behavior from Clickstream Data. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference (LAK 2017)*, March 2017, 21-30. https://doi.org/10.1145/3027385.3027430

[12] Bertrand Schneider, & Paulo Blikstein. 2015. Unraveling Students' Interaction Around a Tangible Interface Using Multimodal Learning Analytics. *Journal of Educational Data Mining* 7, 3, 89-116. DOI: https://doi.org/10.5281/zenodo.3554729

[13] Wanli Xing & Dongping Du. 2019. Throughput Prediction in MOOCs: Using Deep Learning for Personalized Intervention. *J Educ Compt Res* 57, 3, 547-570.

UPreG: An Unsupervised approach for building the Concept Prerequisite Graph

Varun Sabnis
R V College of Engineering, Bangalore
varunsabnis@gmail.com

Kumar Abhinav
Accenture Labs, Bangalore
k.a.abhinav@accenture.com

Venkatesh Subramanian
Accenture Labs, Bangalore
venkatesh.subramania@accenture.com

Alpana Dubey
Accenture Labs, Bangalore
alpana.a.dubey@accenture.com

Padmaraj Bhat
Accenture Labs, Bangalore
padmaraj.bhat@accenture.com

ABSTRACT

Today, there is a vast amount of online material for learners. However, due to the lack of prerequisite information needed to master them, a lot of time is spent in identifying the right learning content for mastering these concepts. A system that captures underlying prerequisites needed for learning different concepts can help improve the quality of learning and can save time for the learners as well. In this work, we propose an unsupervised approach, UPreG, for automatically inferring prerequisite relationships between different concepts using NLP techniques. Our approach involves extracting the concepts from unstructured texts in MOOC (Massively Open Online Courses) course descriptions, measuring semantic relatedness between the concepts and statistically inferring the prerequisite relationships between related concepts. We conducted both qualitative and quantitative studies to validate the effectiveness of our proposed approach. As there are no ground truth labels for these prerequisite relations, we conducted a user study for the evaluation of the prerequisite relations. We build the concept graph using prerequisite relations. We demonstrate few examples of the learning maps generated from the graph. The learning maps provide prerequisite information and learning paths for different concepts.

Keywords

Prerequisite relation, Text mining, Learning path

1. INTRODUCTION

In today's fast-paced world, skill development and a strong foundation in fundamental concepts are becoming very crucial for career growth. MOOCs, offering a wide variety of courses online are becoming ubiquitous among many learners interested in acquiring knowledge and becoming competent in their field of interest. In this journey, learners need to know the order in which they must learn different concepts to attain a good level of mastery in a specific topic. Knowing the prerequisites when learning a topic improves the learning experience of learners and is influential to the learner's achievements [20]. Prerequisite concepts define the concepts one must know or understand first before attempting to learn or understand something new.

With the increasing amount of educational data available, automatic discovery of concept prerequisite relations has become both an emerging research opportunity and an open challenge. There is a growing interest today in researching different techniques for automatically inferring the prerequisite relations between concepts [17][20]. Various solutions like curriculum planning [23], learning assistant [10], automated reading list generation [9] etc, have been developed based on such techniques.

Prerequisites at the course-level have been manually curated by experts and this helps find prerequisite relations between the concepts covered within the courses. For example, concepts in a course on Optimization are prerequisites to concepts in a course on Deep Learning. An example in this scenario would be the Gradient Descent algorithm being a prerequisite for understanding the Backpropagation algorithm used in Deep Neural Networks. Such relations created manually will not scale in real-world online applications. Modern applications today support learning content from a wide variety of domains and cater to learners from multiple educational backgrounds. Manual processes for creating prerequisite relations in such applications are expensive and time-consuming. Hence, it is necessary to develop solutions that can infer prerequisite relations using automated approaches.

In this work, we propose an unsupervised approach, UPreG, for automatically inferring prerequisite relationships between different concepts using NLP techniques. We built a concepts graph capturing the concepts and the prerequisite relation between them. Concepts here refer to technologies, programming languages, tools, and topics in the Software and Computer Science domain. The concepts graph can be

Varun Sabnis, Kumar Abhinav, Venkatesh Subramania, Alpana Dubey and Padmaraj Bhat "UPreG: An Unsupervised approach for building the Concept Prerequisite Graph". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 872-878. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

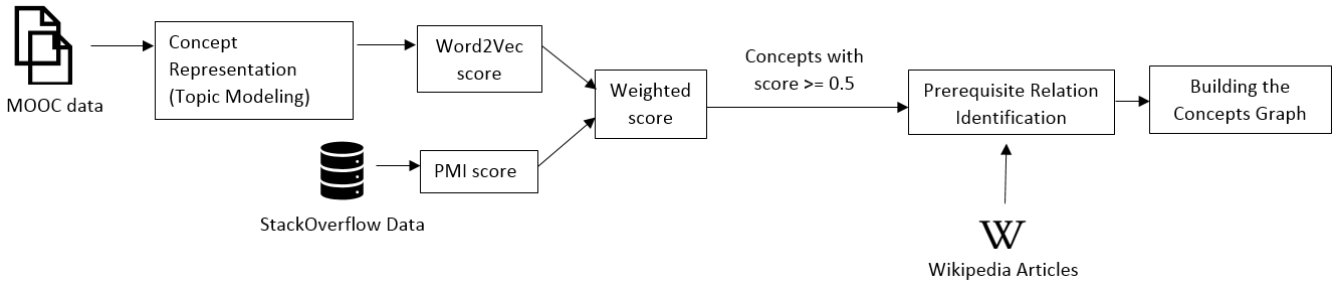


Figure 1: Flow Diagram

leveraged to find the right content for the learner, including the prerequisite content. We conducted both qualitative and quantitative studies to validate the effectiveness of our proposed approach. As there are no ground truth labels for these prerequisite relations, we conducted a user study for the evaluation of the prerequisite relations. We observed that our approach is effectively able to infer the prerequisite relations between concepts. The approach can be extended to other domains as well.

This paper is structured as follows. We present the related work in Section 2. In Section 3, we describe our approach for concept graph generation followed by the evaluation methodology and results in Section 4. In section 5, we discuss the challenges we encountered while building the concepts graph. Finally, Section 6 concludes with future work.

2. RELATED WORK

Pan et al. [17] propose a learning-based method for latent representations of course concepts. They defined various features and trained a classifier that can identify prerequisite relations among concepts. Roy et al. [20] proposed PREREQ, a supervised learning method for inferring concept prerequisite relations. The approach uses latent representations of concepts obtained from the Pairwise Latent Dirichlet Allocation model, and a neural network based architecture. They assumed that concept prerequisites are available to train supervised model. Yu et al. [24] present an improved version PREREQ-S by introducing students' video watch order to enhance the video dependency network. They sorted the watched videos of each student by time and utilize these sequences for replacing the video sequences. They apply two simple DNN models, which first encode the embeddings of the concept pairs and then train an MLP to classify the prerequisite ones. Alzetta et al. [3] applied a deep learning-based approach for prerequisite relation extraction between educational concepts of a textbook. Lu et al. [13] proposed an iterative prerequisite relation learning framework, iPRL, which combines a learning based model and recovery based model to leverage both concept pair features and dependencies among learning materials. Liang et al. [12] addressed the problem of recovering concept prerequisite relations from university course dependencies. They [11] further applied active learning to the concept of prerequisite learning problem. Pal et al. [16] proposed an approach to find the order of concepts from textbooks using the rule-based method. Prior work assumes the prerequisite relationship pairs available as ground truth and apply supervised learning approach. However, acquiring labeled prerequisite pairs is time-consuming and expensive. Currently, the major drawback of supervised learning is that it doesn't perform well over cross-domains [16]. To the best of our knowledge, we are the first to apply unsupervised approach to extract the prerequisite relationship for software domain.

3. APPROACH

In this section, we discuss our approach to build the concepts graph. It is a directional graph where nodes represent the concepts and the edges between nodes represent the prerequisite relationship between them. Our approach in building the concepts graph involves concept representation, measuring semantic similarity between the concepts and identification of the prerequisite relationship between them.

3.1 Concept Representation

The descriptions of the courses in MOOCs contain rich information about the concepts that will be taught to the learners. Many courses do not have annotated course tags to represent the concepts taught in the course. It is very expensive and time-consuming to manually create course tags from the course content [13]. Hence, the concepts must be extracted from the course content using text mining approaches. We collected course metadata from different MOOCs (Udemy and edX) and our internal Learning Management System. We apply Latent Dirichlet Allocation (LDA) [4], a topic modeling algorithm on each course description to extract the concepts. The algorithm generates a topical distribution for each course description. To determine the most relevant topic that represents the concepts a course covers, the topic with highest probability from the distribution is selected. After performing several iterations, we found that setting k=5 (number of topics to be extracted) gave the best results. We extract a total of 9750 unique concepts.

3.2 Semantic similarity between concepts

The Semantic similarity measure between concepts gives a measure of the semantic relatedness between them. Concepts that appear in the same context or appear together very often have higher semantic similarity scores. Semantic Similarity computation eliminates noise present in the results of the topic modeling algorithm and reduces the possibilities of weak relations in the concepts graph. It is also useful in prerequisite relation identification as it is likely that concepts appearing in similar contexts will have better chances of being identified with prerequisite relation. This improves the selection of candidates in the concepts graph.

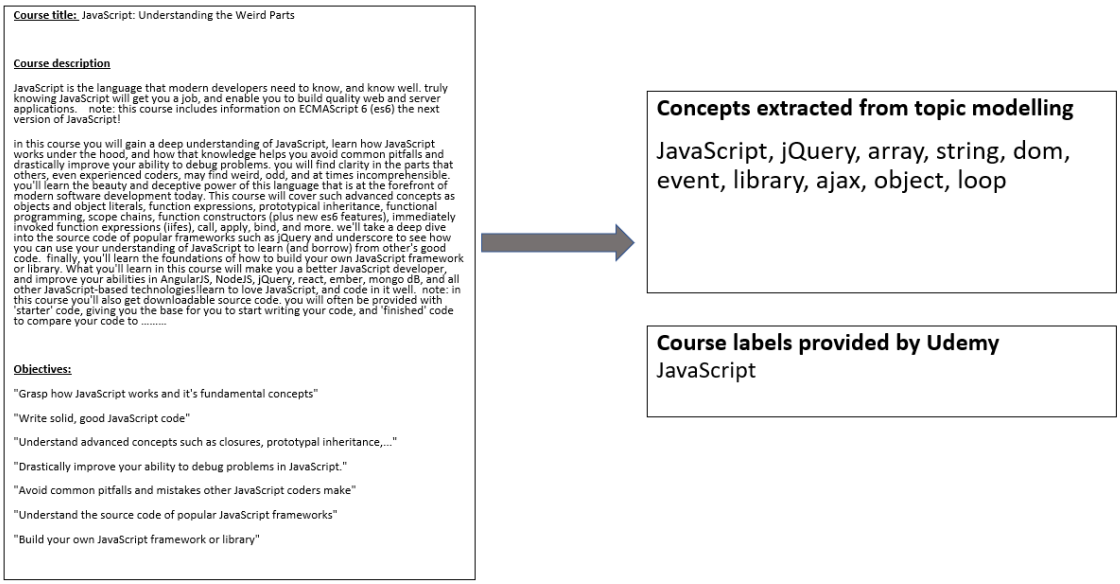


Figure 2: Concepts generated for a JavaScript course

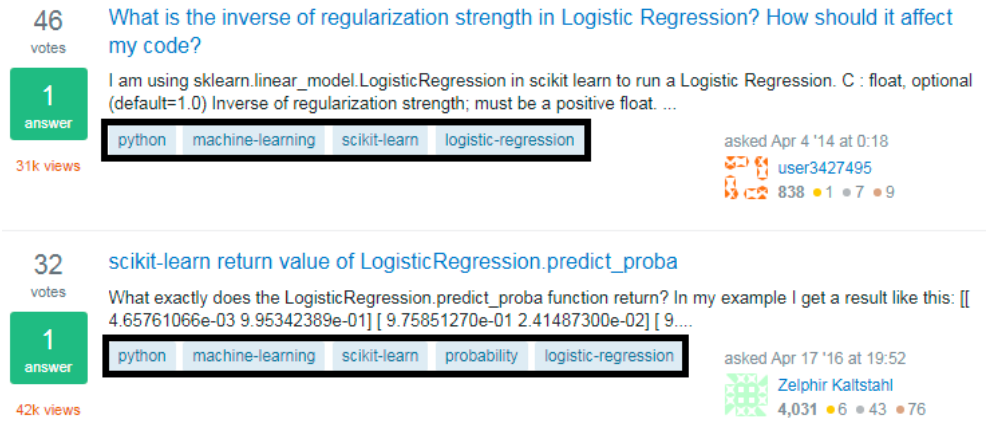


Figure 3: Stack Overflow questions and tags

To measure semantic similarity between the concepts we compute Pointwise Mutual Information (PMI) and Word2Vec cosine similarity scores. The Semantic similarity scores between the concepts are computed as the weighted average of the two scores.

3.2.1 *Pointwise Mutual Information*

PMI gives a measure of concept association used in information theory [6]. It gives a measure of how likely two concepts would occur together when compared to their independent occurrences in the data. For computing the PMI of concept pairs, tags of Stack Overflow questions obtained from Stack Overflow data dumps were used. The author posting a question on Stack Overflow is asked to provide tags associated with the posted question (as shown in Figure 3). Tags that appear often together across all the questions are likely to be strongly related. Higher the score between the two concepts, the more similar they are. We assume that the concepts occurring together have some correlation over a large set of pairs. To compute the PMI scores, we leverage the Stack Overflow dump consisting of 1,000,000 Stack Overflow questions along with their tags [21]. PMI score between any two concepts c_1 and c_2 is defined as:

$$PMI(c_1, c_2) = max\left(0, \frac{\log\left[p(c_1) \cdot p(c_2)\right]}{\log p(c_1, c_2)} - 1\right) \quad (1)$$

Here $p(c_1, c_2)$ is the probability of co-occurrence of concepts c_1 and c_2. It is fraction of Stack Overflow questions in which concepts c_1 and c_2 co-occur as tags. $p(c_1)$ and $p(c_2)$ is the probability of the independent occurrence of concepts c_1 and c_2 as tags across all Stack Overflow questions. The score obtained is a normalized score that takes values between 0 and 1. This ensures PMI and Word2Vec similarity scores have the same scale when taking their weighted average.

3.2.2 *Word2Vec Embeddings*

Raw word frequency is not a great measure of association between words. One problem is that raw frequency is very skewed and not very discriminative. It also does not capture

the kinds of contexts shared between the words, which word embedding techniques capture [2]. We apply Word2Vec approach to learn semantic relatedness between concepts. The Word2Vec model is based on the intuition that words which are similar in context appear closer in the word embedding space. Word2Vec algorithm uses a neural network model to learn word associations from a large corpus of text. We use skip-gram model [15] to learn word embeddings which are low dimensional vector representations of the extracted concepts. The neural network is trained using a text corpus of course descriptions. We train the skip-gram model for generating 300-dimensional word embeddings. Word2Vec neural network is trained using the text corpus of the course description and objectives. We train Word2Vec model on a corpus of 64,150 courses using the Python library gensim [18] with default parameters. Some of the Word2Vec similarity scores between concepts are captured in Table 2. Word2Vec(W2V) similarity score between the concepts is computed as the cosine similarity between these word embeddings.

$$W2V(\mathbf{c_1}, \mathbf{c_2}) = \frac{\mathbf{c_1} \cdot \mathbf{c_2}}{\|\mathbf{c_1}\|\|\mathbf{c_2}\|} \quad (2)$$

Here $\mathbf{c_1}$ and $\mathbf{c_2}$ represent 300 dimensional embedding vectors of concept c_1 and c_2.

Finally, we compute the similarity score as a weighted average of the above two scores. For simplicity, we set the weights to 0.5.

$$Sim(c_1, c_2) = w_1 \cdot W2V(\mathbf{c_1}, \mathbf{c_2}) + w_2 \cdot PMI(c_1, c_2) \quad (3)$$

We observed that extracted concepts can appear with different representations in the Stack Overflow question tags. Examples include synonymous pairs such as node.js and nodejs, javascript and js, mvc and model view controller, etc. To identify such instances, we use the Stack Overflow synonym tag api [22] and identify the matching or synonymous concepts in the Stack Overflow tags. We also filter out irrelevant concepts having no occurrence or synonyms in the Stack Overflow tags. After this process, we end up with 5200 concepts. During the computation of probabilities for PMI scores, we also consider the occurrence count of the synonyms. For example, when computing PMI between javascript and any other concept, we compute the independent and co-occurrence probabilities by counting occurrences of both javascript and js tags in the Stack Overflow questions.

3.3 Identifying Concept Relation

In this section, we explain the process of identifying the prerequisite relationship between different concepts. We only consider the concept pairs with high semantic similarity scores. It is very likely that concept pairs that have very low semantic similarity scores are not related at all and we can ignore such pairs. For example, it is not useful to learn the relationship between Neural Network and PHP which are not related and occur in different domains (deep learning and web development respectively). However, it would be interesting to study the concept pairs Gradient Descent and Backpropagation which are algorithms used in machine learning and share high semantic similarity scores. Inferring the relation that Gradient Descent is a prerequisite of Backpropagation and not vice-versa would be useful. To infer such relations, we make use of Wikipedia articles. For each pair of concepts with high semantic similarity (threshold of 0.5), we compute the concept relevancy scores. For concepts c_1 and c_2, we measure how often the concept c_1 is referred in the Wikipedia article of concept c_2 and vice-versa. Based on the concept relevancy scores, we can infer the prerequisite relation. For example, we know that Java is a prerequisite of Spring Boot. So, it is quite possible that in an explanation for Spring Boot (a Java Web framework), the concept Java would be mentioned more often when compared to the concept Spring Boot being mentioned in an explanation about Java. Algorithm 1 captures the steps to identify the prerequisite relation between concepts.

Algorithm 1 Prerequisite relation inference between concepts

Input: Pair of concepts c_i and c_j which are strongly related, and Wikipedia Knowledge articles.

Output: Relationship between concept pairs (prerequisite relationship) i.e. c_1 is prerequisite of c_2 or vice-versa

1: Tokenize the knowledge articles for all the concepts (C_n), where C_n is set of concepts
2: **for** ordered pair concepts (c_i, c_j) **do**
3: Compute Concept Relevancy scores (CRS) for ordered pairs (c_i, c_j) as

$$CRS(c_i, c_j) = \frac{TF(c_i \in D_j)}{V(D_i, D_j)}$$

$$CRS(c_j, c_i) = \frac{TF(c_j \in D_i)}{V(D_i, D_j)}$$

where c_i and c_j are the concepts for which CRS is computed, D_i and D_j are the wikipedia articles for concepts c_i and c_j respectively, $TF(c_i \in D_j)$ captures the term frequency for concept c_i in wikipedia article D_j, $TF(c_j \in D_i$ captures the term frequency for concept c_j in wikipedia article D_i, and $V(D_i, D_j)$ is the normalization term that captures the total vocabulary in articles D_i and D_j.
4: If $CRS(c_i, c_j) > CRS(c_j, c_i)$, then c_i is prerequisite of c_j and vice-versa
5: **end for**

Table 1: Data collected from online learning platforms

Platform	# Courses	Categories
Udemy	13601	Software development, and design
Edx	1072	Software development
Internal LMS	49202	Software development, and design

3.4 Learning Maps

The identified prerequisite relation pairs were used to build the concept graph. The concept graph has 1325 concepts and 1868 edges. We use networkx [8] python library to build the concept graph. We pass the adjacency list created from the identified concept-prerequisite pairs as an input to the library. The edges in the graph have directions from the concept node to the prerequisite node. The learning maps are built for each concept in the graph using the Depth-first search (DFS) algorithm. They are represented as DFS trees generated by the algorithm. To visualize the learning maps

Table 2: Semantic similarity scores from Word2Vec Embeddings and PMI

c_1	c_2	PMI Scores	W2V Scores
Hadoop	Hive	0.67	0.43
MongoDB	NoSQL	0.64	0.72
JavaScript	jQuery	0.44	0.68
JavaScript	NodeJS	0.57	0.61
Neural Network	Backpropagation	0.74	0.61
Blockchain	Cryptocurrency	0.34	0.73
Inheritance	Polymorphism	0.54	0.62
ASP.NET	Java	0.17	0.08
NodeJS	Promise	0.54	0.21
ASP.NET	C#	0.20	0.42
Hadoop	Java	0.1	0.23
SVM	Classification	0.62	0.43
RDBMS	SQL	0.19	0.37
Machine Learning	Linear Algebra	0.18	0.49

we use d3.js force layout [5]. In visualizing the learning maps we reverse the edge direction, i.e, from prerequisite node to concept node. This is done for the purpose of meaningful and easy identification of prerequisites in the learning maps. The learning maps for the concepts Blockchain and Java Spring framework are shown in Figure 4. The root node colored in blue represents the main concept and all nodes below the root node colored in orange represent the concepts that are prerequisites for the main concept. The child nodes represent the prerequisite concepts for its parent node concept.

Table 3: Extracted prerequisite relation between concepts

c_1	c_2
Distributed systems	Mapreduce
Probability	Logistic Regression
Encryption	Cryptography
Smart Contract	Ethereum
Backpropagation	Neural Networks
Regression	Neural Networks
JavaScript	NodeJS

4. EVALUATION AND RESULTS

4.1 Datasets

We collected metadata about various courses from MOOC platforms and our internal Learning Management System (LMS) using REST APIs. We fetched data from categories relevant to Software Development and Design. The distribution of the number of courses fetched from different platforms is shown in Table 1. There are 13,600 courses from Udemy, 1,050 courses from edX, and 49,500 courses from our LMS in the Software Development and Design category. The output from the REST APIs was in JSON format and each had a different schema. Hence, we selected MongoDB, a NoSQL database to store the retrieved data.

We apply text pre-processing on course metadata. Specifically, the course descriptions from Udemy contain HTML tags. We parse the HTML tags in course descriptions using Beautiful Soup [19]. We remove stopwords and apply Lemmatization and Stemming to reduce words to their base forms. We also create custom stopwords manually by analyzing the topic modeling output. We stored pre-processed data in MongoDB for further processing and evaluation.

4.2 Evaluating extracted concepts

We apply Latent Dirichlet Allocation (LDA), a topic modeling algorithm to infer topics from the course descriptions. We extract five topics from each course description. Each topic is a vector representation that not only indicates the words belonging to the topic but also the probability of the words belonging to the topic. From the topical distribution for the course description, the words from the topic with maximum probability were considered and stored against each course metadata as tags in the database. Figure 2 shows the description and the tags obtained for a Javascript course in Udemy.

To evaluate the concepts extracted from the course description, we apply the Overlap Coefficient to measure the similarity between the concepts extracted from the course description and concepts tagged by Udemy. The overlap coefficient, or Szymkiewicz–Simpson coefficient, is a similarity measure that measures the overlap between two finite sets [1]. It is related to the Jaccard index and is defined as the size of the intersection divided by the smaller of the size of the two sets. Mathematically, we define the Concept overlap coefficient as

$$concept\_overlap(X, Y) = \frac{|X \cap Y|}{min(|X|, |Y|)} \quad (4)$$

where concept_overlap(X, Y) captures the average concept overlap between two sets X and Y, X is the concepts extracted from topic modeling, Y is the concepts tagged in course descriptions of Udemy dataset, and N is the number of course descriptions in the dataset. We observed the average concept overlap coefficient to be 0.97. This shows that the concepts extracted from the topic modeling algorithm quite well capture the relevant concepts covered in the course. Udemy's course description contains a maximum of two concepts tagged. We further analyzed how well our approach is able to identify the other concepts from course descriptions, not captured in Udemy's concepts tag. We performed a quantitative analysis with 20 Subject Matter

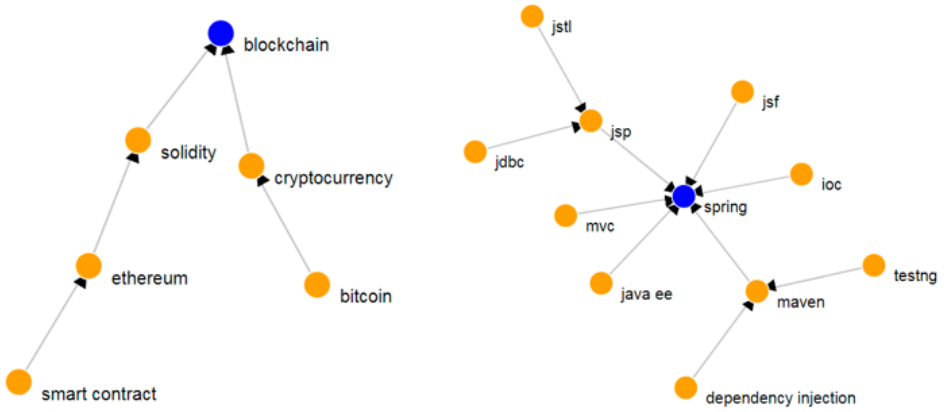


Figure 4: Learning maps for Blockchain and Java Spring framework

Experts (SMEs). The SMEs are having experience ranging from 5-10 years and have worked on different technologies in IT companies. We randomly sampled 100 courses offered on Udemy and provided five courses to each SME along with inferred concepts for each course. The SMEs were asked to provide their response on whether these inferred concepts are relevant for the course or not. We computed the accuracy considering SME's responses as true labels. We observed the accuracy of inferred concepts to be 0.73.

4.3 Evaluating concept Prerequisite Relations

There are no ground truth labels available for inferred prerequisite relationships. To assess the effectiveness of prerequisite pairs generated by our approach, we conducted a quantitative analysis with 25 SMEs to identify if a concept c_1 is a prerequisite for another concept c_2. We created five groups with 5 SMEs in each group. We randomly sampled 250 concept prerequisite pairs. Each group is provided with 50 concept prerequisite pairs. We used the Majority voting approach to aggregate their responses. We computed the accuracy of these pairs considering the SME's response as ground truth labels. We observed the accuracy of concept prerequisite pairs to be 0.82. We also measure inter-rater agreement amongst experts using Fleiss' Kappa [14]. Fleiss' Kappa is a statistical measure for assessing the reliability of agreement between a fixed number of raters when assigning categorical ratings to a number of items or classifying items. If the raters are in complete agreement then $\kappa = 1$. If there is no agreement among the raters (other than what would be expected by chance) then $\kappa \leq 0$. We observed κ coefficient to be 0.74 which indicates a level of strong agreement among the raters. We believe some level of disagreement may be due to the fact that prerequisites can be subjective [12] i.e. it is difficult to get consensus for some pairs of concepts. Different individuals may have different experiences of acquiring knowledge on specific topics, and this may lead to different opinions of the prerequisite requirement for a topic. Some of the extracted prerequisite relationships are shown in Table 3.

5. CHALLENGES

Some of the challenges that we faced while building the concepts graph.

1. For some concepts extracted from the course description we had disambiguation issues when checked in Wikipedia. For example, Java can refer to a programming language or an island in Indonesia. To deal with this issue, we pass the extracted concepts to google search API [7] and fetch the Wikipedia article that is ranked higher in the search results. Due to the popularity of these software concepts, we observe that relevant results were returned by picking the higher ranked Wikipedia article from the search queries.
2. Our inference of prerequisite relationships is based on reference scores computed from Wikipedia articles of the concepts. These scores may not always provide accurate results. It is possible that articles for some of the concepts may have high reference scores for concepts that are derived from it and not vice-versa.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed our approach to infer prerequisite relations between concepts and build the concept graph. The proposed method does not require manually annotated data which was the major drawback of supervised learning approaches. We use relevant data sources in different steps to incorporate relevant and rich semantic information to infer prerequisite relations accurately. To validate our results, we performed both quantitative and qualitative evaluations. The identified concept prerequisite pairs were evaluated by subject matter experts. We observed an accuracy of 0.82 for the inferred prerequisite relations. We built the concept graph from the prerequisite relation pairs and demonstrated few examples of the learning maps generated from the concept graph. Learning maps can be used in many applications ranging from content-based recommendation systems to more sophisticated online tutoring systems etc. As future work, we plan to extend our research by creating a personalized curriculum planner system that captures the concepts learners currently know and what they want to learn. By leveraging this information, the system will create a personalized learning plan for them using their input information and prerequisite relations. Although, our approaches are not limited to the software domain, we plan to carry out further studies and experimentation to measure the system's generalization to other domains.

7. REFERENCES

[1] J. Adler and I. Parmryd. Quantifying colocalization by correlation: the pearson correlation coefficient is superior to the mander's overlap coefficient. *Cytometry Part A*, 77(8):733–742, 2010.

[2] F. Almeida and G. Xexéo. Word embeddings: A survey, 2019.

[3] C. Alzetta, A. Miaschi, G. Adorni, F. Dell'Orletta, F. Koceva, S. Passalacqua, and I. Torre. Prerequisite or not prerequisite? that's the problem! an nlp-based approach for concept prerequisite learning. In *CLiC-it*, 2019.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[5] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec. 2011.

[6] G. Bouma. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40, 2009.

[7] A. Casagrande. Google Search API. https://github.com/abenassi/Google-Search-API, 2020. [Online; accessed 05-Mar-2021].

[8] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[9] J. G. Jardine. Automatically generating reading lists. In *Technical Report UCAM-CL-TR-848*, 02 2014.

[10] L. Jiang, S. Hu, M. Huang, Z. Wang, J. Yang, X. Ye, and W. Zheng. Massistant: A personal knowledge assistant for mooc learners. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 133–138, 2019.

[11] C. Liang, J. Ye, S. Wang, B. Pursel, and C. L. Giles. Investigating active learning for concept prerequisite learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[12] C. Liang, J. Ye, Z. Wu, B. Pursel, and C. Giles. Recovering concept prerequisite relations from university course dependencies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[13] W. Lu, Y. Zhou, J. Yu, and C. Jia. Concept extraction and prerequisite relation learning from educational data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9678–9685, 2019.

[14] M. L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282, 2012.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[16] S. Pal, V. Arora, and P. Goyal. Finding prerequisite relations between concepts using textbook. *arXiv preprint arXiv:2011.10337*, 2020.

[17] L. Pan, C. Li, J. Li, and J. Tang. Prerequisite relation learning for concepts in moocs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1447–1456, 2017.

[18] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

[19] L. Richardson. Beautiful soup documentation. *April*, 2007.

[20] S. Roy, M. Madhyastha, S. Lawrence, and V. Rajan. Inferring concept prerequisite relations from online educational resources. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9589–9594, 2019.

[21] I. Stack Exchange. Stack Exchange Data Dump. https://archive.org/details/stackexchange, 2021. [Online; accessed 05-Mar-2021].

[22] StackExchange. StackExchange. https://api.stackexchange.com/docs/synonyms-by-tags, 2020. [Online; accessed 05-Mar-2021].

[23] Y. Yang, H. Liu, J. Carbonell, and W. Ma. Concept graph learning from educational data. *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pages 159–168, 02 2015.

[24] J. Yu, G. Luo, T. Xiao, Q. Zhong, Y. Wang, J. Luo, C. Wang, L. Hou, J. Li, Z. Liu, et al. Mooccube: A large-scale data repository for nlp applications in moocs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3135–3142, 2020.

Online Estimation of Student Ability and Item Difficulty with Glicko-2 Rating System on Stratified Data

Jaesuk Park
Knowre Korea Inc.
epark@knowre.com

ABSTRACT

We propose an adaptation of the Glicko-2 rating system in a K-12 math learning software setting, where variable time intervals between solution attempts and the stratification of student-item pairings by grade levels necessitate modification of the original model. The discrete-time stochastic process underlying the original system has been modified into a continuous-time process to account for the irregularity of intervals between solution attempts. Also, conceptual prerequisite relationships between items were used to provide initial rating estimates that allow for rating values to be meaningfully compared across grade levels. Fitting the model using real student learning data results in rating value distributions successfully exhibiting a gradation with the increase of grade level. A potential area of application in a personalized education setting is also briefly discussed.

Keywords

Item response theory, dynamic paired comparison model, stratified data, educational assessment, stochastic variance model

1. INTRODUCTION

We consider the problem of assigning appropriate curriculum levels in a large-scale K-12 math learning software to students who are substantially ahead or behind their peers. Previous studies have suggested the importance of matching learning content difficulty to a student's ability for positive student learning outcomes [3, 10, 16]. In light of this, students who are much farther ahead (e.g., gifted students) or behind their peers (e.g., students with learning disabilities) can benefit much from receiving a more tailored educational feedback, based on learner and skill models that can model their differences more effectively.

With the recent advances in computing devices, various approaches have been sought to harness the power of computing to model learners more accurately in educational contexts, as comprehensively overviewed in [2]. In one particular line of approach [11, 13, 12, 15], dynamic paired comparison models were used to quickly estimate student abilities and item difficulties in a scalable manner. In these adaptations, the players consist of students ("users") and units of learning task (e.g., problem items, assignments), and each solution attempt is conceptualized as a match between a student and a learning task, in which the winner earns 1 point and the loser earns 0 points (with no draw). The primary advantage of such models over traditional IRT methodologies is in their ability to compute ability estimates "on the fly" [11] while retaining a similar mathematical structure to IRT.

The problem occurs, however, when the dataset is *stratified*—i.e. when student-problem pairings can be grouped into distinct (or largely nonoverlapping) groups such that a problem's rating cannot be adequately adjusted by a student outside the group to which it belongs. In a K-12 math learning software, because students are only exposed to problems appropriate for their grade level, grade levels serve as strata. Consequently, we cannot adequately tell how a student would perform outside of their regular grade level just by looking at the student's rating value. See Fig. 1 for an illustration.

Ideally, we would not have this problem by gathering enough learning data from a large number of students for 12+ years, during which they would work through all curricula offered by the product in sequence. However, in a commercial educational software context where a user is not bound to use products from just one vendor, this is highly impractical.

Hence we raise a question: is there a way to enforce rating values to reflect the relative positions of the strata, despite the absence of sufficient overlaps in students/items among them? One possible strategy is to initialize the ratings differently for each stratum according to their relative positions, e.g., to initialize first-grade rating values to 100, second-grade rating values to 200, etc., and then let the dynamic paired comparison algorithm do the calibration within each grade level. But then how could we justify that the initial estimation done for all curricula is properly reflective of their actual difficulties relative to one another?

Here, the key insight is that the partial ordering of mathematical concepts due to prerequisite relationships provides a basis for the division of concepts into grade-level curric-

Jaesuk Park "Online Estimation of Student Ability and Item Difficulty with Glicko-2 Rating System on Stratified Data". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 879-885. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

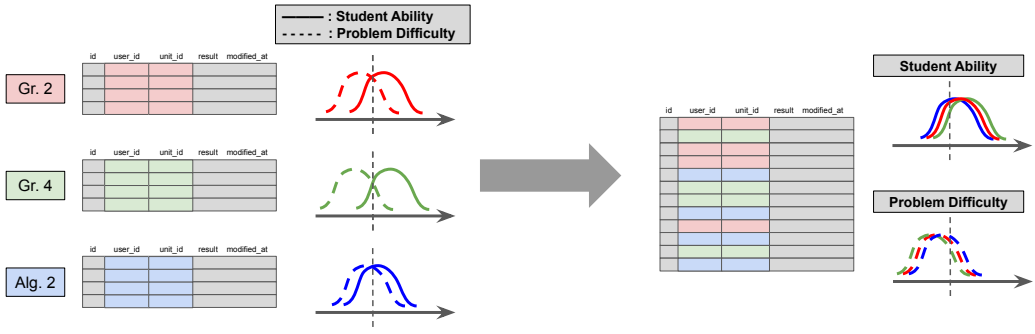


Figure 1: An illustration of the impact of data stratification on the rating interpretability. As a result of stratification, the distributions of rating values can overlap unreasonably much with each other, and the corresponding mean rating values may not align with the actual order of grade levels.

ula, which then in turn stratifies the learning data. In the K-12 math learning software used in our study, each problem item is conceptualized as a particular instantiation of a mathematical concept ("knowledge unit," or just "unit") with specific values. These mathematical concepts have prerequisite relationships defined among them, the collection of which can be represented as a directed graph. We attempt to employ these relationships to obtain statistically interpretable and contextually appropriate estimations.

Specifically, our contribution is twofold: 1) modification of a dynamic paired comparison rating system model to account for imbalance in rating update frequencies between students and items, and 2) use of prerequisite relationships between concepts for rating initialization to achieve rating comparability between curriculum levels. We aim to yield, from a stratified dataset, a set of ratings that can be meaningfully compared across grade levels: where students and items in a lower grade level would generally have lower ratings than those in a higher grade level.

The remainder of this paper is organized as follows. Section 2 presents our particular adaption of a dynamic paired comparison model, including the details for incorporating the conceptual prerequisite information into rating initialization. Section 3 describes the dataset used for evaluating our model and presents our results. Section 4 discusses the potential for applying our model to assign grade levels for students far ahead or behind their peers, lists some of the limitations of our work, and suggests a few possible directions for further research.

2. MODEL

The Glicko-2 rating system [7] falls under the family of dynamic paired comparison models, along with the Glicko rating system [6] (its predecessor) and the Elo rating system [4] (of which the two Glicko systems are extensions). Improving upon its predecessor, the Glicko-2 rating system models the change in variance of player strength as another stochastic process, thereby accounting for the possibility of sudden changes in strength. More specifically, the algorithm models the change in player strength per unit time with a normal distribution with variance equal to the square of the rating *volatility*, whose logarithmic change per unit time is itself normally distributed.

2.1 Continuous-time Glicko-2 Model

The original Glicko-2 system presented in [7] assumes the underlying stochastic processes to be discrete-time, where the overall measurement period is discretized into time increments called "rating periods." Within each rating period, the matches are assumed to occur simultaneously. However, because there is too much imbalance in the average number of matches between users and items, [7]'s recommendation of having 5-10 matches per rating period for every player is not feasible to implement in our application context. [15] has successfully worked around this limitation by constraining each rating period to contain only one match, but the workaround did not account for an increase in rating uncertainty due to the passage of time, which is a key feature of the Glicko rating system family. Here, we take the approach of modifying the Glicko-2 model under a continuous-time stochastic process framework, so that the model can account for rating uncertainty increase due to the passage of time without discretizing the measurement period.

Let $\theta_s(t)$ denote the ability estimate of user s at time t, and let $\beta_i(t)$ denote the difficulty estimate of unit i at time t. Then as a result of using continuous-time stochastic process framework, the model equations for latent trait parameters become

$$\theta_s(t) \sim \mathcal{N}(\mu_s(t),\ \phi_s^2(t)). \tag{1}$$

$$\theta_s(t+\Delta t) \mid \theta_s(t),\ \sigma_s^2(t+\Delta t) \sim \mathcal{N}(\theta_s(t),\ \Delta t\, \sigma_s^2(t+\Delta t)) \tag{2}$$

$$\log \sigma_s^2(t+\Delta t) \mid \log \sigma_s^2(t),\ \tau^2 \sim \mathcal{N}(\log \sigma_s^2(t),\ \tau^2) \tag{3}$$

for user ability estimates, and

$$\beta_i(t) \sim \mathcal{N}(\mu_i(t),\ \phi_i^2(t)) \tag{4}$$

for unit difficulty estimates. Here, as in [8], μ denotes rating, ϕ denotes rating deviation (RD), and σ denotes rating volatility. Note that the difficulty of a mathematical concept is expected to remain constant over time, so we do not impose any stochastic volatility assumption on $\beta_i(t)$.

As for the correctness probability (i.e., the probability of user s correctly answering an instantiation of unit i at time

t), the Glicko rating system family differs from the Elo rating system in its incorporation of rating uncertainty to calculate this quantity. We are generally interested in the correctness probability *before* the user s actually attempts unit i. However, the time elapsed between the user's last attempt and the current attempt can vary throughout the user's activity history, which also varies the amount of inflation to apply each time on the user's rating uncertainty, ϕ_s. Hence we apply equation (2) prior to calculating the correctness probability. Let t_s and t_i denote the last time user and unit latent trait estimates, respectively, were updated. Let $Y_{s,i}(t)$ be a Bernoulli random variable denoting user response correctness. Then the correctness probability is given by:

$$\Pr(Y_{s,i}(t) = 1) = E(\hat{\mu_s}(t),\ \hat{\mu_i}(t),\ \hat{\phi_s^2}(t) + \hat{\phi_i^2}(t)) \quad (5)$$

where $E(\mu_1,\ \mu_2,\ \phi^2) = \left[1 + e^{-g(\phi^2)(\mu_1 - \mu_2)}\right]^{-1}$ is the expected score function that accounts for rating uncertainty [7], and

- $g(\phi^2) = \left[1 + \frac{3\phi^2}{\pi^2}\right]^{-1/2}$,
- $\hat{\phi_s^2}(t) = \phi_s^2(t_s) + (t - t_s)\sigma_s^2(t_s)$,
- $\hat{\phi_i^2}(t) = \phi_i^2(t_i)$,
- $\hat{\mu_s}(t) = \mu_s(t_s)$, and
- $\hat{\mu_i}(t) = \mu_i(t_i)$.

Here, we use $\sigma_s^2(t_s)$ in place of $\sigma_s^2(t)$ to estimate $\hat{\phi_s^2}(t)$, although their equivalence only holds in expectation.

After user s finishes solution attempt for unit i with result $y_{s,i} \in \{0, 1\}$, the update equations for latent trait estimates are given as below, following [7]'s derivation of corresponding equations under the continuous-time framework:

$$\sigma_s^2(t) = \exp\left(\arg\max_{a(t)} p(a(t)|y_{s,i})\right) \quad (6)$$

$$\phi_s^2(t) = \min\left\{\phi_s^2(0), \left[\frac{1}{\phi_s^2(t_s) + \sigma_s^2(t)} + \frac{1}{v_s^2(t)}\right]^{-1}\right\} \quad (7)$$

$$\phi_i^2(t) = \min\left\{\phi_i^2(0), \left[\frac{1}{\phi_i^2(t_i)} + \frac{1}{v_i^2(t)}\right]^{-1}\right\} \quad (8)$$

$$\mu_s(t) = \mu_s(t_s) + \phi_s^2(t) \cdot g(\hat{\phi_i^2}(t)) \cdot (y_{s,i}(t) - E_s(t)) \quad (9)$$

$$\mu_i(t) = \mu_i(t_i) + \phi_i^2(t) \cdot g(\hat{\phi_s^2}(t)) \cdot ((1 - y_{s,i}(t)) - E_i(t)) \quad (10)$$

In these equations, we have

- $E_s(t) = E(\hat{\mu_s}(t), \hat{\mu_i}(t), \hat{\phi_i^2}(t))$,
- $E_i(t) = E(\hat{\mu_i}(t), \hat{\mu_s}(t), \hat{\phi_s^2}(t))$,
- $v_s^2(t) = \left[g(\hat{\phi_i^2}(t))^2 E_s(t)(1 - E_s(t))\right]^{-1}$, and
- $v_i^2(t) = \left[g(\hat{\phi_s^2}(t))^2 E_i(t)(1 - E_i(t))\right]^{-1}$.

Also, in equation (6), $p(a(t)|y_{s,i})$ is the marginal posterior density function for $a(t) = \log \sigma_s^2(t)$, approximated using the product of the following two normal density functions (here, $\varphi(z; m, \varsigma^2)$ denotes the normal density function with mean m and variance ς^2):

1. $\varphi(a(t); a(t_s), \tau^2)$, which comes from equation (3), and
2. $\varphi(\theta_s^*(t); \mu_s(t_s), \phi_s^2(t_s) + (t - t_s)e^{a(t)} + v_s^2(t))$, which is the normal approximation of the marginal likelihood distribution of $\theta_s(t)$, whose mode is denoted with $\theta_s^*(t)$.

The latter normal density function features the quantity $(\theta_s^*(t) - \mu_s(t_s))$, which is approximated in [6] using first-order Taylor expansion.

Finally, note that to prevent a rating deviation from becoming arbitrarily large, the quantity is constrained in equations (7) and (8) to never exceed the value for a brand new user/unit, just like how it was done in [5].

2.2 Initial Parameter Estimation

To address the stratification issue mentioned in the introduction, the user and unit ratings are differentially initialized based on their respective curricula. Instead of setting each curriculum's initial rating value arbitrarily, we want the values to reflect more closely our prior knowledge of the distributions of concepts within each curriculum.

We find this prior knowledge in our proprietary conceptual precedence graph, where units are represented as nodes (vertices) in a directed graph. Each edge (u, v) in the graph is interpreted as: "An instance of unit u is being used as a step in solving an instance of unit v." Hence unit u corresponds to a prerequisite concept that a user must have mastered before being able to successfully master unit v.

The key idea in our usage of the graph is that a question item (corresponding to a specific knowledge unit) that involves one or more steps to solve must in general be harder than any of the steps themselves. Hence we assign each unit with a non-negative integer value, which we call "depth," in such a way that for every edge, the tail node is assigned with a lower depth value than the head node. This way, a concept appearing in a higher grade level would in general correspond to a higher depth value (since they would generally incorporate lower-level curriculum concepts as prerequisites), making the depth values roughly signify how "in-depth" the corresponding concepts are. See Fig. 2 for an illustration.

We also seek to differentiate among units with no parents (i.e., concepts with no prerequisites) by imposing that the depth difference between a unit and its successor be as small in magnitude as possible, while still ensuring that every unit has a strictly greater depth value than any of its parents.

From a graph theory perspective, the problem of assigning depth values can be formulated as a variant of layer assignment problem on a directed acyclic graph $G = (V(G), E(G))$ with minimal dummy vertices, formally stated as the follow-

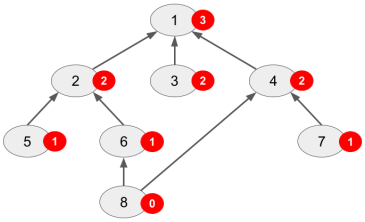


Figure 2: Illustration of assigning depth values to knowledge units in a simple conceptual precedence graph. Knowledge units are represented as nodes (gray ovals). On the right of each oval, a red circle shows the corresponding depth values assigned.

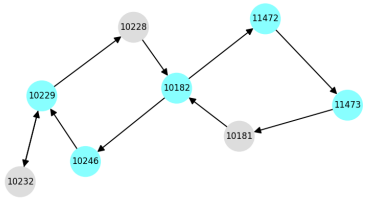


Figure 3: Three instances of simple cycles in the conceptual precedence graph used in our study, which all belong to one strongly connected component. We found that cycles exist mostly due to the presence of "gateway units" (shown in cyan ovals), whose main role is to select which concept to apply from multiple related concepts.

ing integer linear program (ILP):

$$\begin{aligned} \min \quad & \sum_{(u,v)\in E(G)} d(v) - d(u) & \\ \text{s.t.} \quad & d(v) - d(u) \geq 1 & \forall (u,v) \in E(G) \\ & d(v) \in \mathbb{Z}_{\geq 0} & \forall v \in V(G) \end{aligned} \tag{11}$$

(here, $d(v)$ denotes the depth value assigned to node v). For a general overview of the layer assignment problem and its variations, readers are referred to Section 13.3 of [9].

Two challenges arise in initializing rating values through solving the depth assignment problem. The first challenge is that our conceptual precedence graph could contain cycles, such as ones shown in Fig. 3. To address this challenge, we assign the same depth value to all units in the same strongly connected component (SCC), noting that any directed cycle is strongly connected. Implementationally, this corresponds to solving the ILP given in (11) on the conceptual precedence graph's *condensation*, which is a directed acyclic graph formed by contracting each SCC into one node.

The second challenge in assigning depths to nodes on the conceptual precedence graph is that the graph (and thus also its condensation) may consist of multiple weakly connected components (WCCs), which are subgraphs whose underlying undirected graphs are connected. The above ILP assigns depth values relative only to other SCCs in the same WCC, so additional steps must be taken to equate the depth value distributions for each curriculum across all WCCs. In particular, we label each SCC with the lowest-level curriculum that features at least one of its constituent units. Next, we take the smallest number of WCCs that together contain all curriculum labels. We call this collection of WCCs *reference WCCs*. Afterward, we offset the depth value for each SCC in every non-reference WCC to be at least the minimum depth value of all SCCs in the reference WCCs that are labeled with the same curriculum.

Once the adjusted depth values for all SCCs (and thereby all units) are thus computed, each curriculum's depth value is set to be the average depth value of all units in the curriculum.

Below is the summary of procedure for assigning depth $d(k)$ for each curriculum $k \in X = \{1, \ldots, K\}$:

1. Let $G = (V(G), E(G))$ be our conceptual precedence graph, which is a directed graph such that each node $v \in V(G)$ is associated with a curriculum $\chi(v) \in X$.
2. Condense G to yield a directed acyclic graph $C = (V(C), E(C))$.
3. Let $W_1, \ldots, W_n$ be WCCs of C, from largest to smallest.
4. For each $W_i = (V(W_i), E(W_i))$, solve the ILP given in (11) to yield pre-adjustment depth values $d_{init}(S)$ for each SCC S.
5. Label each SCC S with a curriculum
$$\chi_{min}(S) = \min_{v \in V(S)} \chi(v).$$
6. Let $A = \{W_1, \ldots, W_r\}$ be the reference WCCs (defined above), such that r is minimized; i.e., choose no more WCCs than necessary.
7. For each curriculum $k \in X$, let
$$d_{min}(k) = \min\{d(S) \mid \chi_{min}(S) = k, S \in \bigcup_{i=1}^{r} V(W_i)\}.$$
8. For each $W_j = W_{r+1}, \ldots, W_n$, adjust depth value $d(S)$ for each SCC $S \in W_j$ to be at least $d_{min}(\chi_{min}(S))$. However, do so in a way that the adjusted depth values still satisfy the constraints of the ILP given in (11).
9. We now have the adjusted depth values for every SCC $S \in V(C)$. For each SCC S, let $d(v) = d(S)$ for all $v \in S$.
10. For each $k \in X$, let
$$d(k) = \text{mean}\{d(v) \mid v \in V(G),\ \chi(v) = k\}.$$

We now give each user s or unit i associated with curriculum k as follows:

$$\mu_s(0) = \mu_{min} + \alpha \cdot d(k) \tag{12}$$

$$\mu_i(0) = \mu_{min} + \alpha \cdot d(k) \tag{13}$$

where quantities μ_{min} and α are hyperparameters to be optimized.

3. EVALUATION

We evaluate our model using a dataset consisting of student practice records from January 2016 to December 2019 through our adaptive software used in math learning centers located throughout the United States. Students are given problems to practice based on their current grade level and the content areas where they struggle. The data consists of 5,179,493 records of 10,194 users' combined attempts for problems associated with 7,513 knowledge units, ranging from Grade 2 concepts to Algebra 2 concepts. When a student gets a problem wrong in the first attempt, the student gets to make a second attempt for the same problem after being walked through the steps; in our analysis, however, only the first attempt's result was considered.

For the Glicko-2 model hyperparameters, we used the values suggested in [8]: 350.0 for the initial RD (in Glicko-1 scale; [8] shows how to convert between the two scales) and 0.06 for the initial user volatility. In the case of τ, for which a range of values is suggested, we used 0.5. The time elapsed from one attempt to the next, used in rating uncertainty inflation, is measured in days. Finally, through extensive simulations, we chose $\alpha \approx 0.2303$ and $\mu_{min} \approx -2.8782$, which, in Glicko-1 scale (on which the values were originally set), are exactly 40.0 and 1000.0, respectively.

Each unit's associated curriculum was based on the information provided in our content management system. For units appearing in multiple curricula, the earliest curriculum in the sequence was used. For users, due to the lack of availability of exact registration dates for all users at the time of the study, each user's curriculum was set as the curriculum associated with the first unit attempted by the user. The initial parameters for both users and units were then set following the procedure described previously.

3.1 Predictive Performance

To assess the predictive performance of our adaptation of the Glicko-2 rating system, we plotted the change in RMSE values for every 1,000 records over time (for the rationale behind the metric choice, see [14]). As the latent trait estimates are calibrated based on student practice records, we expect the RMSE across the entire system to decay over time. We see that this is exactly the case in Fig. 4, where the calibration curve for our model is also reported along with the reliability and resolution values.

We also report a convergent pattern in unit rating values and dynamically adjusting user rating values, analogous to the results obtained in [15], in Fig. 5.

3.2 Gradation of Unit Rating Distributions

We also plot the distributions of the final unit rating values for each curriculum. We expect that using a conceptual precedence graph to initialize rating values would cause the central tendencies of the rating distributions would show an upward trend as the curriculum level increases. As shown in Fig. 6, the final ratings computed without the graph-based rating initialization fail to show an upward trend in the mean rating values, whereas they do with the graph-based rating initialization. Also noteworthy is the complete disappearance of overlap in IQR between two curricula far

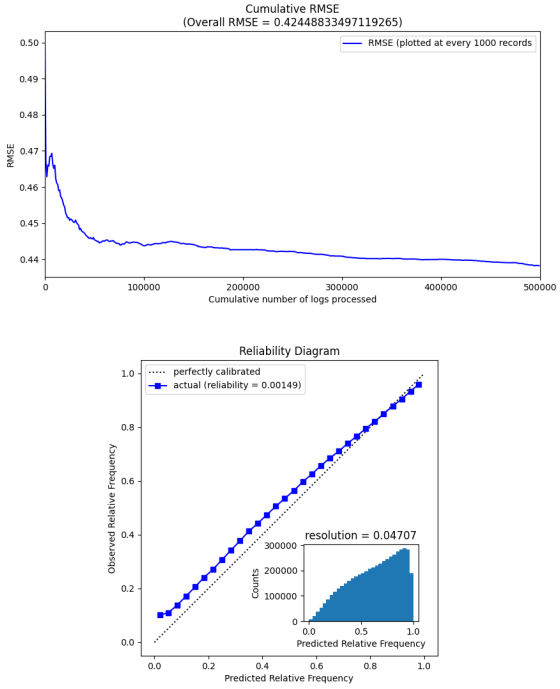


Figure 4: Top: Cumulative RMSE values calculated at every 1,000 records. For effective visualization, only results from the first 500,000 records were plotted. Bottom: Reliability diagram with sharpness graph inserted in the lower right.

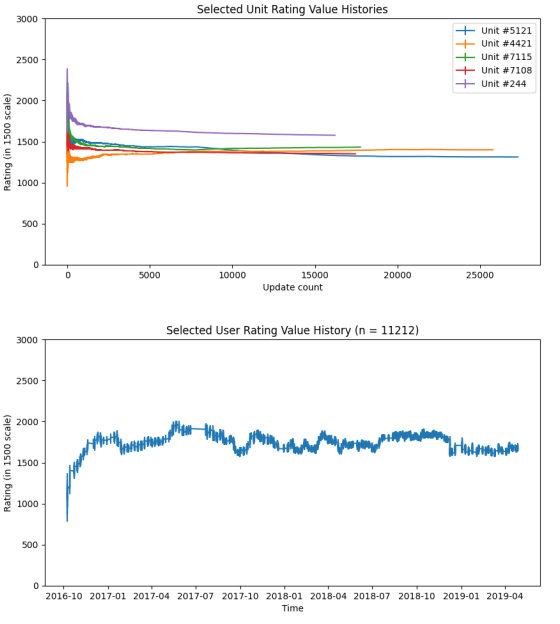


Figure 5: Rating values as a function of time for 5 most frequently attempted units (top) and for the user with the most number of attempts (bottom).

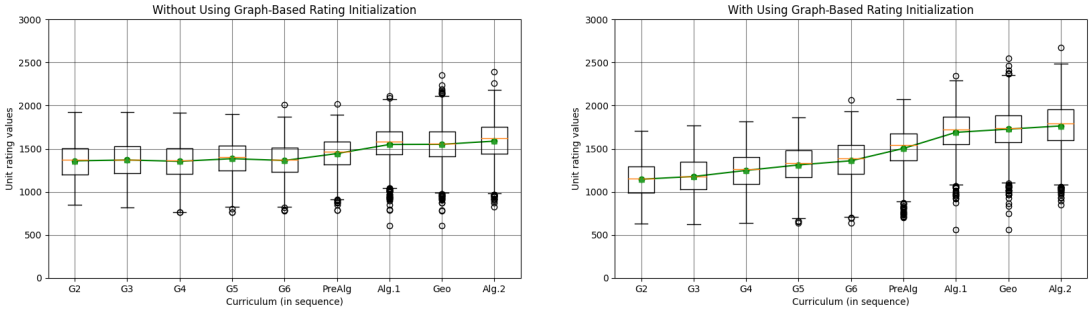


Figure 6: Final distributions of knowledge unit ratings. Orange bars indicate medians, green dots indicate means. Note that the rating values on the vertical axis are on the Glicko-1 scale.

apart from each other, such as Grade 2 and Algebra 2, upon using a conceptual precedence graph to initialize ratings.

4. DISCUSSION

We have used conceptual prerequisite relationships to give our model a better prior distribution— one that better reflects the stratified nature of student practice data. The depth values used to calculate the initial rating values, however, are still quite coarse estimates; for example, the difference in difficulty between a unit and one of its prerequisite units may not be even across the conceptual precedence graph. Nevertheless, we see that the distribution of the lowest-level curriculum (Grade 2 in our study) and that of the highest-level one (Algebra 2 in our study) show a substantially little overlap compared to when we used the initialization method of the original Glicko-2 system, which suggests that there was still a nontrivial improvement. Note that the separation of unit rating distributions between two adjacent curricula (for example, Grade 2 and Grade 3) are not well separated. This is expected, as we would not expect a huge jump in terms of curriculum difficulty from one school year to the next.

One interesting area of application of this framework is determining the appropriate grade level for students whose mathematical achievement levels are substantially ahead or behind their grade levels. With estimates of item difficulties that account for grade-level hierarchy, we can have a data-based justification that would allow gifted students to be placed at a higher-level curriculum that is neither too hard nor too easy for them. Likewise, we could allow for students lagging behind their peers to be placed at a lower-level curriculum, where they could ensure that their foundational understanding of lower-level mathematical concepts is firm before moving onto the next grade level. For this application, a separate round of validation with external measurements, e.g., standardized test scores, must first take place.

A well-known limitation of using the Glicko rating system family for educational applications is its inability to model multiple-choice item correctness probabilities. This is because the correctness probability of such an item has an infimum strictly greater than 0, making the corresponding probability distribution improper. Hence a natural future direction would be to address this limitation, e.g., by incorporating the particle-based method presented in [12].

Another potential threat to the validity of using the Glicko-2 model for student ability measurement is in its unidimensionality assumption. Part of the challenge of verifying whether the student response data can be modeled with a one-dimensional construct in a learning setting is that unlike in IRT settings, a student's ability is expected to change throughout the data collection period. An interesting future direction would be to investigate whether there is sufficient evidence to suggest that students' mathematical ability is multidimensional, and if so, how a model like the Glicko-2 rating system can be extended to reflect the multidimensionality; the degree to which the extension presented in [1] can be applied also remains to be seen.

Also, when assigning each curriculum with a depth value, the average depth values for all constituent units were calculated. In practice, however, as learning software product continues to expand, units can be added or removed, or their edge connections may change. Our current choice of taking an average makes the algorithm sensitive to changes in the conceptual precedence graph's internal connectivity structure. Median may be a more robust, and thus more practical, choice, though this may come at the risk of decreased differentiability across consecutive curricula.

5. CONCLUSION

We have presented an adaptation of the Glicko-2 rating system in a K-12 math learning software context. The stratified nature of student-item pairings has made effective discrimination of students and problems across grade levels challenging. We have shown evidence that by using the prerequisite relationships between concepts to initialize rating values, we can allow for the gradation of rating distributions from lower-level curriculum to the higher-level curriculum while ensuring that the prediction error for student response correctness still decreases over time. A potential area of application is for determining the grade level appropriate for students substantially ahead or behind their peers.

6. ACKNOWLEDGEMENT

I thank my boss Kurt Cho, who gave nudges in productive directions whenever I was stuck, and my colleagues Sunghwan Cho and Seunghun Lee, who worked on developing data pipeline infrastructure on which the proposed model can be deployed. Also, I thank everyone in my company, who patiently waited in support while the project was in the works.

7. REFERENCES

[1] L. Cai. Potential applications of latent variable modeling for the psychometrics of medical simulation. *Military Medicine*, 178(suppl_10):115–120, 2013.

[2] M. C. Desmarais and R. S. J. d. Baker. A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction*, 22(1-2):9–38, 2012.

[3] J. S. Eccles. Expectancies, values and academic behaviors. *Achievement and achievement motives*, pages 74–146, 1983.

[4] A. E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Publishing, 1978.

[5] M. E. Glickman. The Glicko system. `http://www.glicko.net/glicko/glicko.pdf`.

[6] M. E. Glickman. Parameter estimation in large dynamic paired comparison experiments. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 48(3):377–394, 1999.

[7] M. E. Glickman. Dynamic paired comparison models with stochastic variances. *Journal of Applied Statistics*, 28(6):673–689, 2001.

[8] M. E. Glickman. Example of the Glicko-2 system. `http://www.glicko.net/glicko/glicko2.pdf`, 2013.

[9] P. Healy and N. Nikolov. *Hierarchical Drawing Algorithms*, pages 409–454. 08 2013.

[10] C. S. Hulleman, K. E. Barron, J. J. Kosovich, and R. A. Lazowski. Student motivation: Current theories, constructs, and interventions within an expectancy-value framework. In *Psychosocial Skills and School Systems in the 21st Century*, pages 241–278. Springer, 2016.

[11] S. Klinkenberg, M. Straatemeier, and H. L. van der Maas. Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education*, 57(2):1813–1824, 2011.

[12] J. Nižnan, R. Pelánek, and J. Rihák. Student models for prior knowledge estimation. *International Educational Data Mining Society*, 2015.

[13] J. Papousek, R. Pelánek, and V. Stanislav. Adaptive practice of facts in domains with varied prior knowledge. In *Educational Data Mining 2014*, 2014.

[14] R. Pelánek. Metrics for evaluation of student models. *Journal of Educational Data Mining*, 7(2):1–19, 2015.

[15] R. Reddick. Using a Glicko-based algorithm to measure in-course learning. *International Educational Data Mining Society*, 2019.

[16] S. Sampayo-Vargas, C. J. Cope, Z. He, and G. J. Byrne. The effectiveness of adaptive difficulty adjustments on students' motivation and learning in an educational computer game. *Computers & Education*, 69:452–462, 2013.