Towards the Prediction of Semantic Complexity Based on Concept Graphs

Rémi Venant, Mathieu d'Aquin

Towards the Prediction of Semantic Complexity Based on Concept Graphs

Rémi Venant
Le Mans University
LIUM - EA 4023, Le Mans Université
72085 Le Mans, Cedex 9, France
remi.venant@univ-lemans.fr

Mathieu d'Aquin
Insight Centre for Data Analytics
National University of Ireland
Galway
mathieu.daquin@insight-centre.org

ABSTRACT

The evaluation of text complexity is an important topic in education. While this objective has been addressed by approaches using lexical and syntactic analysis for decades, semantic complexity is less common, and the recent research works that tackle this question rely on machine learning algorithms that are hardly explainable and are not specifically designed to measure this variable. To address this issue, we explore in this paper the engineering of novel features to evaluate conceptual complexity. Through the construction of a knowledge graph that captures the concepts present in a text and their generalized forms, we measure different graph-based metrics to express such a complexity. Eventually, early-stage evaluations based on a well-known public corpus of students' productions show that the use of these metrics significantly improves performance compared to a state-of-the-art binary neural network classifier.

Keywords

semantic complexity, concept complexity, knowledge graph, features engineering, neural network, machine learning

1. INTRODUCTION

In Technology Enhanced Learning, the evaluation of the complexity of textual material underlies several activities. For instance, assessments in language classes are based, among other things, on the measurement of the learners' abilities to deal with different grammatical structures or to use the most precise vocable to express their thoughts. Another example can be found in learning resources indexation. Merlo [4] and Openstax[1] are two projects that aim at offering fine-grain information retrieval functionalities and automatic recommendation systems for educational objects based on their metadata, including their level of difficulty.

Complexity is usually related to readability, a concept defined as the "total sum of all those elements within a given piece of printed material that affect the success a group of readers have with it" [5]. Thus, readability depends on both the object (the text) and the subject (the reader), whereas complexity is commonly characterized by a function whose output does not differ from one reader to another [11, 21].

Many research works in Natural Language Processing (NLP) sought out a way to measure the complexity of a text, and to predict the category (e.g. the level of difficulty) it should fall into. Most of them, nevertheless, focus on lexical and syntactic evaluation to achieve their objectives. Alongside the recent progress in machine learning, and more specifically in deep learning, another approach to text classification arouse, based on semantic relationship of words within a text [17]. While offering outstanding performance, these predictive models are not easily explainable [28]. However, to provide this property is of importance to increase the confidence a user gives to these systems [1], and would allow to improve the usability of predictive models. For instance, a tutoring system designed for learning a foreign language would benefit from such a model, that is able not only to assess the complexity of a student's writing, but also to give suggestions on how to improve it.

Hence we propose in this paper a novel approach to measure semantic complexity with a model based on explainable features: we exploit the semantic web to build the conceptual representation of a text within an ontological graph, that we use to compute a set of metrics. In order to validate our model, we focus here on the following research questions:

- Is a model based on our sole metrics able to outperform a state-of-the-art model of semantic complexity?
- Does the extension of a state-of-the-art predictive model with our engineered features improves its performance?

In this paper, we answer these questions in the context of complexity assessment of students' productions in English as a foreign language. The first section is dedicated to related work in complexity assessment, in order to select a state-of-the-art model to compare to. We describe then the pipeline we designed to build a conceptual graph from a text and convert it into a vector, before going into the details of the 17 metrics that compose the vector. The fourth section is dedicated to the analysis of our model, in order to answer the research questions we defined previously.

[1] https://openstax.org

Rémi Venant and Mathieu d'Aquin "Towards the prediction of semantic complexity based on concept graphs" In: *Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019)*, Collin F. Lynch, Agathe Merceron, Michel Desmarais, & Roger Nkambou (eds.) 2019, pp. 188 - 197

2. RELATED WORK

Complexity, as a function of the text only [29], is involved in different tasks related to education. Classification of resource materials has become an important topic with the growth of open education and MOOCs [15], in order to provide suitable recommendations to a learner [3]. Learner evaluation would also benefit from such a function. Checking mistakes is not enough to assess the writing skill of a student; one also needs to measure the ability to express more complex thoughts, with the use of precise concepts [19]. A last instance of a task that requires complexity assessment is text simplification, a process that aims at providing a simpler version of a text by reducing its complexity, without removing its substantial content [25].

These objectives require computational models in order to classify them or to measure a value of complexity according to the needs. Different categories of metrics are involved to build such models, and most of the current research works rely on surface, syntactic and lexical features [8, 7, 24]. Surface level features provide basic statistical measures such as the number of characters or words. Lexical features target the structure of sentences, such as the average number of verb phrases per sentences or the number of dependent clauses [21]. Finally, syntactic features are based on the recognition of terms to compute metrics such as the average number of synonyms of words. Other morphological values (i.e., based on the structure of text) can be found. Siddharthan [29] proposed to measure discourse complexity to evaluate whether connections between text segments are vague or weak. In addition, Davoodi [8] used coherence features, that refer to the grammatical and lexical links which connect linguistic entities together, in order to include the influence of discourse structure on text complexity assessment. Similar features can be found in [24], who evaluated the lexical cohesion and the level of argumentation.

With these features, predictive models of complexity have already shown good performances. For instance, a SVM binary classifier using 117 parameters that belong to these categories of features achieved to classify Swedish texts according to their complexity with an accuracy of 98.9% [12]. Although widely used, surface and syntactic features work only on the structure of the text, while lexical features, using a base of knowledge for word recognition do not provide any semantic to these words. Thus, a few projects proposed different ways to add semantic information to their model. A good example of the different approaches to semantic complexity can be found in [6], who proposed a mixed model using a wide range of different operators, a few of them being related to semantics. Indeed, they used Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), and a Word2Vec model to extract semantic features.

LSA [9] and LDA [2] are traditional machine learning techniques well-known for their efficiency to extract the topics of a text. Thus, a predictive model of complexity based on either LSA or LDA will detect the main topics of a text and use them to assess complexity. This method is then based on the assumption that some topic are more complex than others. There are, however, two disadvantages with this assumption. First, the model will fit the distribution of topics among the texts from the dataset used to train and test it, which can be detrimental to its generalizability. Also, we assume here the texts that deal with the same topics are of the same complexity, an hypothesis that may be wrong.

In a different way, words embedding models such as Word2Vec [23] or GloVe [27] learn geometrical encodings of words from their co-occurence information. Both model achieve to capture the semantics of "analogy". For instance, computing the difference between vectors of words "king" and "queen", then adding the vector of "princess" would give a result whose the closest known vector would be the one of the word "prince". Both models perform well on different kinds of tasks, such as semantic relatedness (to predict the degree of semantic similarity between two words) or concept categorization. These models seem also to provide the best results in capturing semantic complexity [17].

Unfortunately, these techniques project words into an abstract linear space, mathematically meaningful, but hardly understandable. If we can have insights of the vectors' relationships at the scale of words, as with the example given previously, their interpretations regarding complexity remains limited. At the scale of a text, where we usually compute the average vector of words, we do not know any method to interpret the resulting vector regarding the text complexity.

3. CONCEPTUAL GRAPH PIPELINE

Within the field of semantic analysis, the manipulation of concepts within a text is inherent to its complexity [18]. For instance, when concepts are numerous, abstracts, or not closely related to each other, readers may suffer from accessing their prior knowledge to understand the text [10]. We propose here a concept-based approach of feature engineering to assess semantic complexity.

An ontology (or knowledge graph) is a powerful model to represent concepts and their relationships. With the growth of research in the semantic web, cross-domain description of the world became available, and one of the most known nowadays is DBpedia[2]. DBPedia is a crowd-sourced project to provide an open knowledge graph based on the information available in several Wikimedia[3] projects. It provides thus the description in a structured and linked way of various concepts (e.g.: persons, places, organizations, movies, etc.). The English version describes more than 4 million entities so far, and localized versions are provided in 125 languages. Also, everything described in DBPedia is an entity structured through several ontologies (e.g., the DBPedia ontology, schema.org or YAGO). Our work is based on the exploitation of DBPedia and its ontologies to (i) capture conceptual entities from a text, (ii) build a concept graph that includes entities and their higher-order concepts, and (iii) transform the graph into a vector of features. We designed a pipeline to achieve these tasks, whose implementation in Python is open source and publicly available[4]. The pipeline, shown in Figure 1, is built over 5 main components: (i) a text preprocessor, (ii) an entity extractor, (iii) a concept enhancer, (iv) a graph builder and (v) a graph vectorizer.

[2]https://wiki.dbpedia.org/
[3]https://www.wikimedia.org/
[4]https://github.com/afel-project/pySemanticComplexity

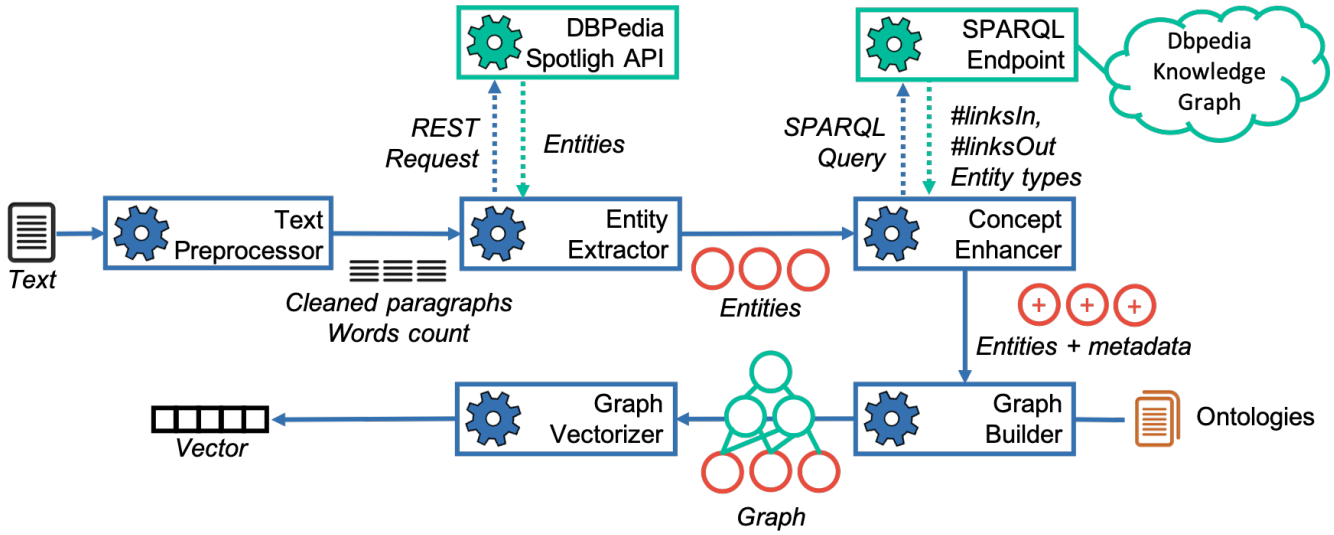


Figure 1: Text to Concept Vector Pipeline.

These software components can be launched separately, either in a stream or batch mode, to support distributing computing. Before we present them in the following subsections, we need to clarify the vocabulary we use to deals with concepts and concept graphs. A concept extracted from a raw text is named **entity**, to follow the vocabulary of DBpedia. An entity is related to some **types**, which are its direct higher-order concepts. Types are classes defined in the ontologies. A **higher-order concept** is a more abstract concept (i.e., a generalization of the concept). Within the ontologies, multiple inheritance relations exist between classes, and allow to structure the concepts they express in term of orders of abstraction. For instance, the "1965 ford mustang" is an entity with a type "collection car", whose one of the higher-order concept is "car", generalized itself into "vehicle", then finally "object".

3.1 Text preprocessor

Given a text document in a unicode format, this component (i) cleans it, with the deletion of forbidden characters, such as null or backspace, then (ii) splits the text into paragraphs (a prerequisite for the entity extractor that cannot process long text). The text preprocessor also computes the number of words (used for some of the metrics at the end of the pipeline) and the offset for each paragraph (used to locate entities within the whole text).

3.2 Entity Extractor

The entity extractor aims at extracting DBpedia entities from a given list of paragraphs. In order to extract such entities from a text, this component interacts with DBPedia through the REST interface exposed by DBPedia Spotlight[5], a tool that performs named entity recognition [22]. Given a text, and quality parameters (i.e., promience, topical pertinence contextual ambiguity and disambiguation confidence), the service returns a list of DBPedia entity URIs with their position in the text and some quality metrics. Thus, from a list of paragraphs, this pipeline stage retrieves a set of DBPedia URI with their positions in the document.

[5]https://www.dbpedia-spotlight.org/

3.3 Concept Enhancer

This stage takes a list of entities and, for each of them, retrieve (i) its related types (second-order concepts), (ii) the number of entities that point to it (#linksIn) and (iii) the number of entities it points to (#linksOut). As we explained before, the entities in DBPedia are represented in a knowledge graph, where entities are linked each other, and described through different ontologies. #inLinks and #outLinks are computed based on the relationship between entities in DBpedia.

In order to retrieve all these types for the list of entities, the component interrogates a SPARQL Endpoint. SPARQL (recursive acronym which means SPARQL Protocol and RDF Query Language) is a SQL-like query language to retrieve or manipulate data in the RDF (Resource Description Framework) format, used in DBPedia. The concept enhancer fetches in a first request all the types that are related to each entity of the list. Two others requests are achieved to compute #linksIn and #linksOut for each entity. Finally this stage returns the list of entities enhanced with their types and the two basic metrics.

3.4 Graph Builder

The role of the graph builder is twofold: (i) to retrieve the higher-order concepts related to types given along with the entities, and (ii) to build an acyclic graph of all concepts.

The higher order concepts are the super-classes of the types that have been previously retrieved. Indeed, the ontologies used in DBpedia provide a hierarchical structure of classes, where each class may have one or several parent classes. In other words, a concept C may have one or several parents (higher-order concepts), that generalize C and its sibling.

In order to retrieve theses higher-order concepts, the graph builder does not need to interact with any DBPedia end-

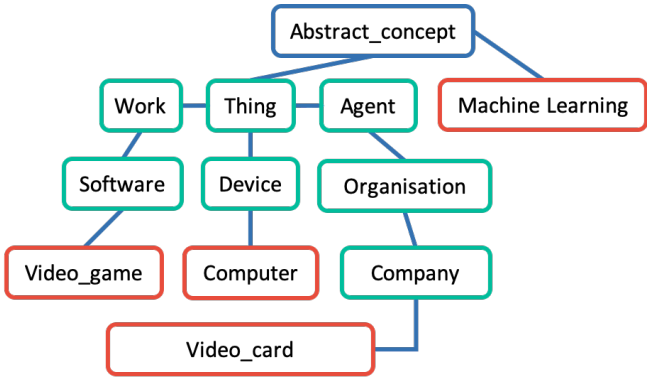


Figure 2: Example of a Concept Graph.

point. This component loads directly the structure of ontologies, which are RDF files that can be locally processed. In our pipeline, we exploit the three most used ontologies in DBPedia: the DBPedia ontology, schema.org and YAGO. For each given type that belongs to one of the three ontologies, the graph builder recursively retrieves its parent concepts. As a result, the component constructs a set of pairs <child concept, parent concept> and, with the given list of entities, builds a concept graph of the text.

This graph is an acyclic graph composed of two kinds of vertices: entities (that appear in the text) and higher-order concepts (classes from the ontologies). An entity vertex has the following attributes: (i) the DBpedia URI, (ii) the list of positions in the text (word index) (iii) the #linksIn value and (iv) the #linksOut value.

While an entity may appear several times in the text, it is represented as a single vertex in the graph. However, its different positions are recorded in its attributes. The other vertices (higher-order concepts) contain only their URI. In order to ensure the graph is connected (a mathematical property required to compute some of the features explained in the following section), an abstract highest-order vertex is inserted and linked to any vertex that does not have any higher-order concept.

We illustrate in Figure 2, an example of a graph computed from the text "This computer has a powerful graphic card that is suitable for video games and machine learning.". Entity nodes are colored in red, while higher-order concepts are in green (the abstract highest-order concept is in blue). In this example, the Spotlight API extracted 4 entities from the DBpedia ontology: "video game", "computer", "video card" and "machine learning". The green nodes on the graph are then classes of this ontology that generalize the concepts they are connected to. Thus, a video game is a software, which is a work, whose generalized concept is a thing.

3.5 Graph Vectorizer

Giving the graph, and the number of words computed in the first stage of the pipeline, this last component applies several calculus in parallel to produce 17 metrics. The values are embedded in an vector, which is the final output of the pipeline. This vector can be used afterwards as an input for a predictive model on conceptual complexity.

4. CONCEPT COMPLEXITY METRICS

In this section, we explain the metrics that highlight specific information about the conceptual complexity, based on the structure of the generated graph. The 17 metrics computed by the pipeline are the following: (i) *#concepts*, (ii) *#distinctConcepts*, (iii) *conceptsByWords*, (iv) *distinctConceptsByWords*, (v) μ*Types*, (vi) σ*Types*, (vii) μ*linksIn*, (viii) σ*linksIn*, (ix) μ*LinksOut*, (x) σ*LinksOut*, (xi) *#nodes*, (xii) *Radius*, (xiii) *Diameter*, (xiv) *Density*, (xv) *Assortativity*, (xvi) μ*TextDensity* and (xvi) σ*TextDensity*. We details them in the following subsections.

4.1 Basic Concept Metrics

The first four metrics are measures based on the entities extracted from the text. **#concepts** is the total count of concepts the entity extractors retrieved, while **#distinctConcepts** is the number of the different concepts extracted. These metrics are based on the assumption that the more concepts a text deals with, the more complex it is.

Since these features might be correlated to the size of the text (which, as explained later, was not the case in our experiments, but still might be), we also propose **conceptsByWords** and **distinctConceptsByWords**, the ratio between #concepts and #distinctConcepts respectively, and the number of words.

4.2 Concept Connection Metrics

The six following indicators relate the direct properties of the concepts that appear in the text. μ**Types** and σ**Types** are respectively the mean and standard deviation of the number of types per entity. As explained before, each concept extracted from the text is linked to its types (i.e. its direct higher-order concepts). We suppose here that the more an entity has types, the more concepts of higher-order are required to explain it, and thus the more complex this entity is. We use the two basic statistic descriptors to capture that notion at the scale of a document.

For each entity in our graph e, we also have *#linksIn*, the number of links that go from entities of the global DBPedia knowledge graph to e, and *#linksOut*, the number of links that go from e to others entities of that same graph. We compute then two statistic descriptors for each indicator at the document level. μ**linkIn**, σ**linkIn** are respectively the mean and standard deviation of the number of entities in DBPedia that point to the entities of the document, while μ**linkOut** and σ**linkOut** are about the entities in DBPedia that are pointed by the entities of the document. We suppose here that the more relations an entity has with others, the more popular it is, and the less complex it might be.

4.3 Concept Abstraction Metrics

The next five metrics take into account the whole concept graph: the entities, their types and the higher-order concepts retrieved recursively from the ontologies.

#nodes is the total number of nodes in the graph. The higher it is, the more concepts have been used or the more specific they are (i.e., the more higher-order concepts there are to specify them).

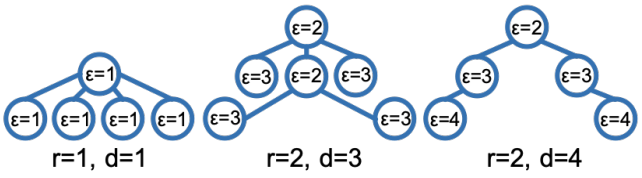


Figure 3: Examples of radius and diameters.

For the next metrics a formalism is required. Let $G = (V, E)$ the definition of a graph G with V the set of vertices (the concepts) and E the set of edges. Let $\forall(u, v) \in V^2,\ d(u, v)$ the distance between the nodes u and v: the number of edges in the shortest path. Since our graph is connected (thanks to the abstract higher-order concept node) and undirected, we have the following properties:

$$\forall(u, v) \in V^2,\ d(u, v) \geq 1, d(u, v) = d(v, u) \tag{1}$$

Interpreted in our context, the distance between 2 concepts shows how close they are from each other. Since concepts are linked by their higher-order concepts, the distance is the length of the path from one concept c_1 to another concept c_2, going through their closest common higher-order concept. Thus, the distance measures how much we have to generalize to find the common inherent concept to relate c_1 and c_2.

ϵ is the eccentricity of a vertex v. It is defined as the greatest distance between v and any other vertex of the graph.

$$\forall v \in V,\ \epsilon(v) = \max_{u \in V} d(v, u) \tag{2}$$

In our context, the eccentricity for a concept c_1 gives a measure of how far this concept is from the others.

4.3.1 Radius and Diameter

Based on this eccentricity, the **radius** r_G is the minimum eccentricity of any vertex in the graph.

$$r_G = min_{v \in V} \epsilon(v) = min_{v \in V}[max_{u \in V} d(v, u)] \tag{3}$$

At the opposite, the **diameter** of a graph d_G is the maximum eccentricity of any vertex in the graph.

$$d_G = max_{v \in V} \epsilon(v) = max_{v \in V}[max_{u \in V} d(v, u)] \tag{4}$$

On the one hand, the diameter highlights the "spreadness" of concepts: the more unrelated and specific concepts we have, the higher the diameter will be. On the other hand, the radius points to the "compactness" of concepts out: the more the concepts are closely related to each other, the lower the radius will be. Note that the radius is not strictly the opposite of the diameter; both metrics measure information that are not theoretically linearly dependent. To have an insight of their meaning, Figure 3 shows different simple graph structures with their radius and diameter.

4.3.2 Density

The **density** of a graph stresses how much nodes are connected to each other. A graph is dense if the number of edges is close to the maximal possible number of edges. The opposite is a sparse graph, that has few edges. The density is computed by the following equation:

$$0 \leq \mathcal{D}_G = \frac{2|E|}{|V|(|V| - 1)} \leq 1 \tag{5}$$

Here, a concept graph with a high density implies that concepts and higher-order concepts are closely related to each other. The text may deal with many different concepts, but many of them share parents concepts and so on. Density may then be a factor of discrimination regarding two texts that have similar numbers of concepts, but with one using concepts from a unique domain while the other one deals with varied concepts.

4.3.3 Assortativity

Following Newman [26], the **assortativity** measures in a graph the similarity of connections with respect to the vertice degree. The degree is the number of connections (the number of edges) a vertex has to other vertices. Mathematically, the assortativity is the Pearson correlation coefficient of degree between all pairs of vertices. It is computed by the following equation:

$$\mathcal{A}_G = \frac{2\sum_i j_i k_i - \frac{1}{2}|E|^{-1}(\sum_i j_i + k_i)^2}{\sum_i (j_i^2 + k_i^2) - \frac{1}{2}|E|^{-1}(\sum_i j_i + k_i)^2} \tag{6}$$

where j_i, k_i are the degrees of the vertices at the ends of the i^{th} edge, in a graph with $|E|$ edges.

Because $\mathcal{A}_G$ is a correlation coefficient, it lies between -1 and 1. When it is close to 1, the graph shows a perfect assortative mixing: the vertices in the network that have many connections tend to be connected to the other vertices with many connections. When $\mathcal{A}_G$ is close to -1, the graph is disassortative: the nodes that have many connections tend to be linked to the nodes with few connections. When $\mathcal{A}_G$ is close to 0, the graph is non assortative: there is no particular correlation between node connections and their degree.

Because in our concept graphs, concepts (vertices) are connected only by their relationship of abstraction (a parent concept being a more general concepts), a positive assortativity would signify that the more parents a concept requires to be defined, the more grand-parents these parents have. Under the hypothesis that the more higher-order concepts we need to define a concept c, the more complex c is, the assortativity may be a candidate metric to evaluate complexity: a graph with a high assortativity may highlight a text that deals with very complex and precise concepts.

Compared to the μTypes indicator, which is only computed on the basis of entity types, this metric takes into account the whole graph. For instance, if the text uses very narrowed concepts, that have only few types, which are however defined by many higher-order concepts, μTypes would be low, but assortativity may be close to 1.

4.4 Concept Organization Metrics

While the previous metrics are about the graph only, they do not take into account the positions of the different entities

within the text. Although they provide conceptual information about the overall document, they lack giving insight about the evolution of complexity at a local level. For instance, a paragraph that deals with closely related concepts may be simpler than a paragraph that manipulates different unrelated concepts. Thus two texts that handle similar concepts, but with a different structure, may have a different complexity. In order to capture such information, we have built a metric based both on the concept graph and on the positions of the entities within the text.

Let $N \subset V$ the subset of vertices that contains the entities. As we explained in the previous section, the entities include the list of the positions of their occurrences. Let $occ(e)_i \; \forall e \in N$ the position of the i^{th} occurrence of the entity e in the text. We define the function sp of two entities that returns the shortest path in the text between the two concepts.

$$\forall (n,m) \in N^2, \; sp(n,m) = \min_{i,j} |occ(n)_i - occ(m)_j| \quad (7)$$

On the basis of sp, we define the textual density td between two entities as the product of their graph distance d and the opposite of their textual distance td, normalized by the product of the graph diameter D_G and the text length L_T.

$$\forall (n,m) \in N^2, \; td(n,m) = \frac{d(n,m)}{D_G} \cdot \frac{1 - sp(n,m)}{L_T - 1} \quad (8)$$

Since $0 \leq d(n,m) \leq D_G$ and $0 \leq sp(n,m) \leq L_T - 1$, we have $0 \leq td(n,m) \leq 1$. When td is near 0, the concepts are either semantically very close, or unrelated and far to each other in the text. When td is close to 1, the concepts are semantically far to each other but appear closely in the text. At the document scale, we compute then the two last metrics μ**TextDensity** and σ**TextDensity** that are respectively the mean and standard deviation of the textual density of all pairs of entities.

5. CONCEPTUAL COMPLEXITY ASSESSMENT EVALUATION

In this section, we seek to evaluate the performance of a classifier that predicts complexity, based on the concept metrics defined above. In order to analyze the impact of our features on such predictive models, we carried out several rounds of evaluation. We started building a binary classifier based on the state-of-the-art features to obtain a model of reference. To answer our first research question, "Could a model based on our sole metrics outperform a state-of-the-art predictive model of semantic complexity?", we trained another classifier that uses our conceptual complexity features only and compared it with the first one.

Afterwards, we considered the second question: "Does the extension of a state-of-the-art predictive model with our engineered features improves its performance?". To compare two models, one being an extension of the other, we trained our two models on the same splits of data (within a cross-validation procedure) and compared the two lists of measures of performance with a statistical test to assess whether the performances of the extended model was significantly higher than the performances of the first one.

Finally, since results were positive with respect to question 2, we trained a last classifier based on syntactic, lexical and semantics features, to evaluate how well it could perform for the specific task of predicting the overall complexity of learner's productions.

5.1 Dataset

All models here were trained on an extract of the EF-Cambridge Open Language Database [16, 13]. This database is a text corpus of documents written by adult learners of English as a foreign language. In this study, we use a subset of this database, that includes 41 626 essays.

Human examiners evaluated these essays in order to assess the learners' level of knowledge defined in the global scale provided in the Common European Framework of Reference for Languages[6]. This scale define 6 levels of knowledge (i.e.: A1, A2, B1, B2, C1 and C2), that describe skills in reading, listening, speaking and writing. Regarding this last domain of competency, A1 and A2 levels target beginners: they can interact in a simple way, using familiar everyday expressions and very basic phrases. Independent users belong to the B level group. Compared to level groups A and C, this group presents a clear difference between its two levels. While B1 users can produce simple connected texts on topics that are familiar, B2 users are able to write clear and detailed text on a wide range of topics and give their point of view on a topical issue. Finally, C1 and C2 levels target proficient users, that can handle complex subjects and produce clear, well-structured and detailed texts.

Thus, in this dataset, examiners labeled learners' essays with one of these levels. Although there is no formal description about the evaluation process, the definitions provided in the global scale of the framework relate to the different kind of complexity we exposed earlier (i.e.: syntactic, lexical and semantic). For this study, we define two categories of writings based on these labels. The documents evaluated with a level between A1 to B1 are considered in the G1 group, while the documents evaluated with a level between B2 to C2 belong to the G2 group.

Outliers were removed from the dataset. They were defined here as the documents where the number of words were below the first centile or above the last one, or where the number of concepts were again, below the first or above the last centile. Since the distribution of documents between G1 and G2 was skewed, we reduce the dataset to obtain an equal proportion of samples in each group. The baseline score of a binary classifier is then 0.5 for accuracy, precision or recall.

Also, we detected a potential bias in the dataset, as the samples in G2 tend to be longer (in numbers of words) than the ones in G1. To prevent our models from reproducing that bias, we filtered all conceptual features we explained previously that would present a significant correla-

[6]https://www.coe.int/en/web/common-european-framework-reference-languages/level-descriptions

tion ($r \geq 0.75$, $p\text{-}value < 0.003$) with the length of the text (p-value was computed using the Bonferroni correction since we computed different statistical tests, which raise then the risk to produce a statistical type I error). It appears that none of the 17 features were significantly correlated to the length of the text as much as this threshold. Finally, we examine the potential relationship between pairs of features, to assess their independence and to avoid training our models with correlated features. We looked for strong significant correlations: $r \geq 0.9$ with $p\text{-}value \leq 0.0004$ (again, p-value was computed using the Bonferroni correction). It appears that only #concepts and #distinctConcepts are strongly correlated ($r = 0.936$). We then decided to remove the latest feature from our models for this dataset.

5.2 Baseline Model Evaluation

In order to produce a baseline binary classifier of semantic complexity, we applied the following methodology: (1) compute a 300 dimensional average vector for each sample based on pre-trained Glove vectors ; (2) train different types of classifiers using 5-folds cross validation, with default hyperparameters for each of them; (3) select the model that provides the best f1 score; (4) split the whole dataset into one subset for hyper parameters tuning (90%), and one subset for final testing (10%); (5) tune the hyper parameters of this model using a grid search 10-folds cross validation and (7) test the final model on the last subset.

Glove is an unsupervised machine learning algorithm used to compute vector representation for words, proposed by Stanford University [27]. Based on the word-to-word co-occurence statistics obtained from a corpus, GloVe computes a representation of words into a vector space. In our case, we did not train such a model from scratch on our dataset, but used pre-trained words vectors available online [7]. These vectors were computed from the 2014 corpus of Wikipedia and the 5th edition of English Gigaword [8], in a 300 dimensions space. For each sample from our dataset, we computed the mean of the tokenkized words vectors, using 0 padded vectors for unknown words. The tokenization process was achieved using the NIST Tokenizer[9].

The main scoring metric we used to evaluate our models is the f1-score. This metric takes in account both precision and recall through the following formula: $2 \cdot P \cdot R/(P + R)$, given the precision P (the ratio between the number of true positives and the sum of the number of true positives and false positives) and the recall R (the ratio between the number of true positives and the the sum of the number of true positives and false negatives). This metric is a value between 0 (the worst) and 1 (the best).

In step 2, we tested 10 different models. Each model was a pipeline made of two stages: a pre-processing stage to standardize features by removing the mean and scaling to unit variance, and the classifier stage. The results for the classifiers selection are shown in Table 1. The best classifier was the multilayer perceptron (MLP); it achieved the best f1 score, but also the best average accuracy. MLP is a type of feedforward neural networks composed of at least one hidden layer of nodes, where each node in the hidden and output layers uses a nonlinear activation function [14].

Table 1: Glove-based Models Performances.

Classifier	μF1	μAccuracy
MLP (1 hidden layer of size 100)	0.936	0.935
SVC with rbf kernel	0.936	0.934
K-nearest neighbors	0.917	0.910
SVC with polynomial kernel (d=2)	0.916	0.911
Quadratic discriminant	0.907	0.905
SVC with linear kernel	0.899	0.897
Random Forest (100 estimators)	0.896	0.892
AdaBoost	0.865	0.862
Gaussian Naive Bayes	0.826	0.811
Decision Tree	0.811	0.810

In the last step, we tuned the following hyperparameters of the MLP model, given with their set of evaluated values: **(i)** the activation function (logistic, hyperbolic tangent or Rectified Linear Unit), **(ii)** the regularization term α (0.0001, 0.001, 0.01, 0.1 or 1.0), **(iii)** the size of each hidden layer (100 or 200), **(iv)** the number of hidden layers (1 to 5), **(v)** the learning rate (constant or adaptive) and **(vi)** the solver used for the weight optimization (adam or lbfgs).

We tested the different combinations of hyperparameters with a grid search 10-folds cross validation on a stratified subset of 90% of our dataset, using the mean test f1 value as the scoring method. The best configuration appears to be a MLP with one hidden layer of 200 nodes using a ReLU (Rectified Linear Unit) activation function, an α value of 0.01 with an adaptive learning rate and an adam solver.

Table 2: Test Scores of the Baseline and Concept Based Models.

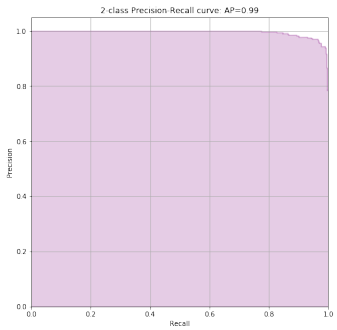
Score	Baseline M. Values	Concept M. Values
F1	0.937	0.888
Accuracy	0.939	0.889
Precision	0.982	0.894
Recall	0.896	0.882

Eventually, we tested that classifier on the remaining 10% of our dataset. The different metrics are exposed in the column "Baseline M. Values" of Table 2. The Precision-Recall curve is illustrated in Figure 4a. Overall, this baseline classifier shows better precision than recall. As we can see on Figure 4a, recall tends to drop quickly as precision goes over 0.9. Recall is defined as the ability for the classifier to avoid false negatives. In our context, recall is thus the ability for the model to avoid predicting a text as being simple while it is in reality complex. In other terms, this models is better at predicting a text as being complex, than predicting a text as being simple.

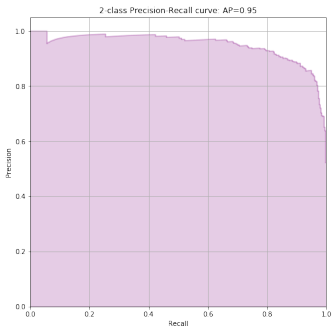
5.3 Concept Based Model Evaluation

On the basis of the pipeline architecture elicited previously, we trained and tested an equivalent model that works with our 16 elicited engineered features only. The model is then composed of a standardization stage and an MLP classifier

[7] http://nlp.stanford.edu/data/glove.6B.zip
[8] https://catalog.ldc.upenn.edu/LDC2011T07
[9] http://www.nltk.org/_modules/nltk/tokenize/nist.html



(a) Baseline Model.



(b) Concept Features-Only Model.

Figure 4: Precision-Recall Curves of Baseline and Concept Features-Only Models.

with the previous selected hyperparameters. This model was trained in a stratified 10-folds cross validation on 90% of the dataset and tested with the remaining 10%. The results are given in the third column of Table 2. The Precision-Recall curve is illustrated in Figure 4b.

Using only the conceptual graph-based features, we can see that this model is worst than the baseline one to predict complexity. The drop of score value is around 5% for F1 and accuracy, and 8% for the precision. In conclusion, to answer to our first research question, using the features we proposed only, we cannot outperform a model based on the state-of-the-art semantic features to predict complexity.

5.4 Baseline and Mixed Model Comparison

Whereas the model based on conceptual features failed to outperform the baseline classifier, these metrics might still add information that are not present from the state-of-the-art semantic features. Thus, we need to study whether a classifier would do better with both sets of features.

In order to assess such a result, and because the two classifiers will not be independent, we need to evaluate whether the performance scores of these classifiers have a significant difference of mean. We trained two classifiers, CL_{base} and CL_{mixed} with the same architecture than the one selected previously: a standardization pre-processing stage chained to a MLP classifier with the former chosen hyperparameters. CL_{base} is the baseline classifier that uses the Glove features only, while CL_{mixed} uses both Glove and our conceptual graph-based features. In order to compare the difference of performances, we evaluated our classifiers with the same splits of a stratified 10-fold cross validation scheme.

The results for the 4 metrics (F1, accuracy, precision and recall) for CL_{base} and CL_{mixed} on each fold are given in Table 3. The metrics observed for CL_{mixed} seem to be better than CL_{base}. However, to assess whether this difference is significant, we proceed a paired sample Student's T-test, since the measurements of each classifier were applied on the same splits of data.

The Table 4 presents the statistical results of the paired T-test. The null hypothesis is that the pairwise difference between the two tests for each metrics is equal. With the degrees of freedom $df = 9$ (as we achieved 10 measures), and a p-value of 0.05, the t-table value is 1.812. For each metrics, we can see on Table 4 than the computed t-value has a absolute value above 1.812. We can then rejected the null hypothesis that there is no difference between means for each metrics. In conclusion, adding conceptual graph-based features to Glove features improve significantly the performance scores of our complexity classifier. We therefore answer positively to our second research question.

5.5 Extended Complexity Classifier

Thus, conceptual graph-based complexity features can increase the performances of complexity classifiers, while improving the explainability of the model. In our context of evaluating learners' production in English to assess their complexity, we eventually would like to test how a complete classifier, using not only semantic features but also syntactic and lexical ones, would perform.

In that last part, we trained and tested a supervised classifier that predicts complexity based on the previous semantic features, and syntactic and lexical features proposed in [20], that appears to be the state-of-the-art metrics so far. We removed the ones strongly correlated with the length of the text, as our dataset presents a potential bias of categorical distribution over the text length. We ended up with 30 syntactic and lexical features, 300 features from Glove and 17 conceptual features. The classifier has the same architecture as those trained before.

The results for the test are presented in table 5, while the Precision-Recall curve is illustrated in Figure 5. With high scores in both precision and recall, this model seems to be suitable to assist teachers in their assessment of complexity.

6. CONCLUSION AND PERSPECTIVES

At the core of several learning related activities, the assessment of complexity is a task of importance. This topic has been broadly studied through the structural analysis of texts, with the design and evaluation of lexical, syntactic or morphological measures. The consideration of semantic metrics, however, is still scarce. Recently, word embedding techniques have demonstrated their promising potential to design powerful predictive models of semantic complexity, but lack explainability. In order to overcome this obstacle, we proposed an approach based on the exploitation of existing knowledge graphs to generate a graph representa-

Table 3: Comparative Scores of Baseline and Mixed Classifiers.

CL_{base} scores				CL_{mixed} (Mixed) scores			
F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.
0.943	0.943	0.940	0.947	0.954	0.954	0.958	0.949
0.950	0.950	0.946	0.955	0.962	0.962	0.966	0.957
0.938	0.937	0.926	0.951	0.947	0.946	0.934	0.960
0.947	0.947	0.937	0.957	0.952	0.951	0.945	0.959
0.947	0.947	0.933	0.963	0.953	0.953	0.952	0.953
0.941	0.940	0.930	0.952	0.946	0.946	0.937	0.956
0.944	0.945	0.945	0.944	0.955	0.955	0.951	0.959
0.947	0.947	0.939	0.955	0.958	0.958	0.956	0.960
0.954	0.954	0.952	0.956	0.965	0.965	0.965	0.964
0.941	0.940	0.920	0.964	0.956	0.955	0.947	0.964

Table 4: Paired T-test Results between Scores of Baseline and Mixed Classifiers.

Scores	Computed t-value
F1	-9.230
Accuracy	-9.271
Precision	-6.303
Recall	-1.922

Table 5: Test Scores of the Extended Complexity Classifier.

Score	Value
F1	0.970
Accuracy	0.969
Precision	0.956
Recall	0.984

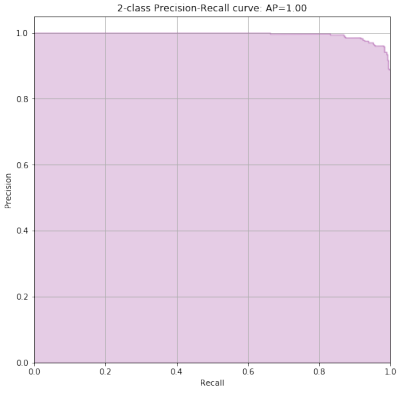


Figure 5: Precision-Recall Curve of the Extended Complexity Classifier.

tion of a text, whose concepts are exposed and related each other through their abstractions. We suggested 17 metrics computed from this concept graph to highlight intelligible information about the semantic complexity of the text.

We evaluated our proposition within the context of learners' productions for a European certification of English as a foreign language. We observed that a word embedding based classifier still tends to surpass a model relying solely on our features. Nevertheless, a classifier that uses these two sets of features together outperforms significantly the previous ones. At last, we proposed a classifier using these semantic metrics but also syntactic and lexical features. Tested in our context of learners' evaluation, its performances seem to make it suitable to assist human examiners in their tasks.

Eventually, we found, at the time of writing, a similar approach to ours, based on the DBPedia knowledge graph to measure conceptual complexity [30]. Authors used concepts extraction in the context of text simplification. Although not based on a concept graph as we did, the metrics they proposed may be also relevant for our domains of application. We will then integrate them to our model shortly. While first results obtained are promising, we have to dig into the analysis of each metric we suggested, in order to evaluate their power of discrimination regarding the conceptual complexity. This work will allow us to validate the interpretation of the features we proposed. At a longer term, it will offer opportunities to consider a proactive usage on users to assist them in their learning activities. Assessing complexity is also of importance for learning resources indexing, to provide useful recommendations. Since this domain of application is different from the present context of study, we will consolidate a dataset of learning objects and then reproduce the experimentation to evaluate how our approach generalizes to other tasks.

7. REFERENCES

[1] O. Biran and K. McKeown. Human-Centric Justification of Machine Learning Predictions. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 1461–1467, California, 2017. International Joint Conferences on Artificial Intelligence Organization.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[3] V. Butoianu, P. Vidal, E. Duval, J. Broisin, and K. Verbert. User Context and Personalized Learning: a Federation of Contextualized Attention Metadata. *Journal of Universal Computer Science*, 16(16):2252–2271, 2010.

[4] C. Cechinel, S. S. Alonso, M.-Á. Sicilia, and M. C. de Mattos. Descriptive Analysis of Learning Object Material Types in MERLOT. In *Research Conference on Metadata and Semantic Research*, pages 331–341. Springer, 2010.

[5] E. Dale, J. C. E. English, and 1949. The concept of readability. *Elementary English*, 26(1):19–26, 1949.

[6] M. Dascalu, P. Dessus, S. Trausan-Matu, M. Bianco, and A. Nardy. ReaderBench, an Environment for Analyzing Text Complexity and Reading Strategies. In *International Conference on Artificial Intelligence in Education*, pages 379–388. Springer, 2013.

[7] M. Dascalu, G.-M. Gutu, S. Ruseti, I. C. Paraschiv, P. Dessus, D. S. McNamara, S. A. Crossley, and S. Trausan-Matu. ReaderBench - A Multi-lingual Framework for Analyzing Text Complexity. In *European Conference on Technology Enhanced Learning*, pages 495–499. Springer, 2017.

[8] E. Davoodi and L. Kosseim. On the Contribution of Discourse Structure on Text Complexity Assessment. *arXiv.org*, Aug. 2017.

[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] C. A. Denton, M. Enos, M. J. York, D. J. Francis, M. A. Barnes, P. A. Kulesz, J. M. Fletcher, and S. Carter. Text-Processing Differences in Adolescent Adequate and Poor Comprehenders Reading Accessible and Challenging Narrative and Informational Text. *Reading Research Quarterly*, 50(4):393–416, Oct. 2015.

[11] J. Falkenjack and A. Jönsson. Classifying easy-to-read texts without parsing. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 114–122, 2014.

[12] J. Falkenjack, K. H. Mühlenbock, A. J. P. o. t. 19th, and 2013. Features indicating readability in Swedish text. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, pages 27–40, 2013.

[13] J. Geertzen, T. Alexopoulou, and A. Korhonen. Automatic linguistic annotation of large scale l2 databases: The ef-cambridge open language database (efcamdat). In *Proceedings of the 31st Second Language Research Forum. Somerville, MA: Cascadilla Proceedings Project*, 2013.

[14] 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.

[15] Q. Han and F. Gao. Towards semantic learning object metadata: mapping standard metadata specifications to ontologies. In *Proceedings of IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE) 2012*, pages H1C–12. IEEE, 2012.

[16] Y. Huang, A. Murakami, T. Alexopoulou, and A. Korhonen. Dependency parsing of learner English. *International Journal of Corpus Linguistics*, 23(1):28–54, May 2018.

[17] Z. H. Kilimci and S. Akyokus. Deep Learning- and Word Embedding-Based Heterogeneous Classifier Ensembles for Text Classification. *Complexity*, 2018(7):1–10, 2018.

[18] W. Kintsch, T. A. Van Dijk Psychological review, and 1978. Toward a model of text comprehension and production. *Psychological review*, 85(5):363, 1978.

[19] B. Kopainsky, P. P. Dummer, and S. M. Alessi. Automated assessment of learners' understanding in complex dynamic systems. *System Dynamics Review*, 28(2):131–156, Apr. 2012.

[20] X. Lu. Automatic analysis of syntactic complexity in second language writing. *International journal of corpus linguistics*, 15(4):474–496, 2010.

[21] X. Lu. Automated measurement of syntactic complexity in corpus-based l2 writing research and implications for writing assessment. *Language Testing*, 34(4):493–511, 2017.

[22] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM, 2011.

[23] T. Mikolov, K. C. 0010, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013.

[24] D. Napolitano, K. Sheehan, and R. Mundkowsky. Online Readability and Text Complexity Analysis with TextEvaluator. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 96–100, 2015.

[25] S. Narayan and C. Gardent. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 435–445, 2014.

[26] M. E. Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

[27] 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.

[28] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?". In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[29] A. Siddharthan. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298, 2014.

[30] S. Štajner and I. Hulpus. Automatic assessment of conceptual text complexity using knowledge graphs. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 318–330, 2018.