Traitement du langage naturel appliqué à la représentation de textes narratifs par réseaux de personnage

Arthur Amalvy

AVIGNON UNIVERSITÉ

THÈSE DE DOCTORAT D'AVIGNON UNIVERSITÉ

École Doctorale n°536
Agrosciences et Sciences

Mention de doctorat :
Informatique

Laboratoire Informatique d'Avignon

Présentée par
Arthur Amalvy

Natural Language Processing for the Representation of Narrative Texts through Character Networks

---

Soutenue publiquement le 9 Décembre 2024 devant le jury composé de :

Claire Gardent	DR	Université de Lorraine, CNRS/LORIA	Rapporteuse, Présidente
Christophe Cerisara	Chargé de Recherche HDR	Université de Lorraine, CNRS/LORIA	Rapporteur
Farah Benamara	PR	Université Paul Sabatier, CNRS/IRIT	Examinatrice
David Bamman	AP	UC Berkeley, School of Information	Examinateur
Vincent Labatut	MCF HDR	Avignon Université, LIA	Directeur de thèse
Richard Dufour	PR	Nantes Université, LS2N	Co-directeur de thèse

Natural Language Processing for the Representation of Narrative Texts Through Character Networks

Traitement du langage naturel appliqué à la représentation de textes narratifs par réseaux de personnage

Arthur Amalvy
Supervised by: Vincent Labatut and Richard Dufour

ACKNOWLEDGMENTS

A thesis is only the emerged part of a long journey and, as most journeys, is marked by the meeting of many people along the road. The first persons I met on this adventure are, of course, my advisors Vincent and Richard. They have been with me at every step, introducing me to the world of research and providing me with guidance. This alone would have already qualified them to be great advisors, but I must add that they also possess deeply human qualities and have been compassionate and understanding when I needed it the most (because, of course, one always needs it at some point in a perilous expedition). For all of these reasons, and the unstoppable joking at the beginning of each of our meetings, I thank you both deeply, and I hope that we can find occasions to work together again in the future.

Then, I met all the wonderful people in the lab. The "PCCA" gang (no, you will not find this one in the list of acronyms of this thesis) and our Street Fighter sessions, Juan and our shared passion for science fiction, Philippe with whom I participated multiple times to the France Robotics Cup, Jarod and our hackerspace project, all of the PhD students, and many others.

During this adventure, I also encountered colleagues across scientific domains and physical frontiers, and I am grateful for our fruitful collaborations. We worked with Pádraig, Shane and Maddy on *A Song of Ice and Fire*, with Professor Kou, Hannah, Hayley and Manuel on *Hunger Games*, and with Nicolas on *Lorenzaccio*. I also recall my meeting with Tanzir, whom I met by a stroke of astronomical luck in the middle of several thousands persons at EMNLP 2023. Thank you for that crazy moment where you walked me through your implementation of the Smith–Waterman algorithm with your laptop in the middle of the welcome party.

The last persons I met are the members of the jury who accepted to review my thesis: Claire Gardent, Christophe Cerisara, Farah Benamara and David Bamman (although I must say that Vincent and I were lucky to see a great talk from Professor Bamman at ACL 2023 prior to the defense). Thank you all for taking the time to review this work, for taking interest in it, and for the lively and interesting discussions we had during the defense.

Some people were also here from the beginning. My family, of course: my brave mother, my always calm father, my energetic sister, my supportive aunt Boubou and uncle Éric, my grandparents Patrice and Danièle. But also, my great friends from UTBM and the ME chat (they will recognize themselves).

Finally, the friends that know me well will wonder if I haven't forgotten someone. And I haven't! An important partner in all of this adventure was the wonderful GNU Emacs software. Emacs is amazing, and extensible, and yes I couldn't stop myself from mentioning it in the acknowledgments. In general, I want to thank all of the free software programmers whose programs I used throughout the thesis: Their thankless efforts make the world run.

RÉSUMÉ

Un réseau de personnages représente des personnages comme des sommets dans un graphe, et leurs relations comme les arêtes entre ces sommets. Dans le cas des oeuvres littéraires, ils permettent de modéliser un récit entier en utilisant un seul objet mathématique. En fonction des besoins, leurs arêtes peuvent représenter différents types d'interactions : co-occurrence, conversation, action directe... De plus, les changements temporels dans les relations peuvent être modélisés avec des réseaux dynamiques. Grâce à cette flexibilité, les réseaux de personnages ont été utilisés pour s'attaquer à plusieurs tâches, comme la classification de genre littéraire, la segmentation de récit, la recommandation ou le résumé automatique. Extraire ces réseaux manuellement est cependant coûteux, et de nombreux chercheurs sont donc intéressés par l'automatisation de ce processus. Cette automatisation nécessite de résoudre différentes tâches de traitement du langage naturel telles que la reconnaissance d'entités nommées (REN), la résolution de coréférences ou l'attribution de locuteur.

Dans cette thèse, nous présentons des contributions à ce processus d'extraction automatique dans le cas des romans, ainsi qu'à des applications des réseaux de personnages. Nous proposons Renard, un pipeline d'extraction modulaire que nous mettons à disposition sous une licence libre. Nous l'utilisons pour mieux comprendre la performance des pipelines existants en étudiant l'impact des erreurs de REN et de résolution de coréférences sur la qualité des réseaux extraits. Nous observons que la performance des deux tâches est importante, et dépend fortement du roman étudié. Pour la résolution de coréférences, nous notons également que l'impact dépend du type d'erreur : la précision des liens de coréférence extraits est particulièrement importante afin de détecter des personnages.

En outre, nous identifions et contribuons à deux défis des systèmes d'extraction de réseaux de personnages. Le premier est le manque de données littéraires pour entraîner ces systèmes. Nous nous nous y attaquons 1) en proposant un nouveau jeu de données littéraire couvrant la REN et la résolution d'alias et 2) en proposant d'utiliser une technique d'augmentation de données, le remplacement de mentions, dans le cas de la REN inter-domaines. Le second défi que nous identifions est la portée limitée des modèles à base de transformers, qui peut être préjudiciable à la performance de certaines tâches. Nous proposons de récupérer du contexte pertinent au niveau du document pour atténuer le manque d'information induit par cette faible portée, et montrons que cela peut augmenter la performance de la tâche de REN.

Enfin, nous présentons des contributions aux applications des réseaux de personnages dans le cadre de deux études de cas. Premièrement, nous utilisons des réseaux modélisant différents types d'interactions dans une analyse de *Lorenzaccio* d'Alfred de Musset. En utilisant la détection de communautés, nous identifions les intrigues de la pièce, quantifions leurs importances relatives et déterminons les interactions entre elles. De plus, nous proposons une méthode automatique pour détecter des conspirations. Deuxièmement, nous proposons d'employer les réseaux de personnages pour résoudre la tâche d'alignement narratif sur trois adaptations du *Trône de Fer* de George R. R. Martin : les romans originaux, les comics adaptés de ceux-ci et la série télévisée. Nos résultats montrent que les méthodes basées sur les réseaux peuvent être meilleures que celles basées sur le texte, et peuvent être combinées avec ces dernières pour améliorer la performance. Nous mettons aussi en valeur l'importance de réaliser la tâche d'alignement sur des unités narratives commensurables. Dans ces deux études de cas, nous montrons l'intérêt des réseaux dynamiques.

ABSTRACT

A character network represents characters as vertices in a graph, and their relationships as edges between them. In the case of literary works, they model a whole narrative using a single mathematical object. Depending on the needs, their edges can represent different types of interactions between characters: co-occurrence, conversation, direct action... Additionally, the temporal changes in the relationships between characters can be modeled with dynamic networks. Thanks to this flexibility, character networks have been used to tackle a number of tasks, such as literary genre classification, story segmentation, recommendation or summarization. Manually extracting these networks is costly, which is why many researchers interested in automating the process. This, in turn, requires solving different Natural Language Processing (NLP) tasks such as Named Entity Recognition (NER), coreference resolution or speaker attribution.

In this thesis, we present contributions to this automatic extraction process in the case of novels, as well as to character network applications. Inspired by the 2019 survey of Labatut and Bost that summarizes existing extraction efforts in a generic extraction framework, we propose Renard, a modular character network extraction pipeline that we release under a free license. We use Renard to better understand the performance of existing extraction pipelines by studying the impact of NER and coreference resolution errors on the quality of extracted networks. We find that both tasks' performance is important to network quality and depends strongly on the novel. In the case of coreference resolution, we also observe that different errors do not have the same impact: linking precision is particularly important when it comes to correctly detecting characters.

Additionally, we identify and work on two challenges of automatic character network extraction systems. The first one is the lack of literary data to train such systems. We tackle this challenge by 1) releasing a new literary dataset covering the NER and character unification tasks; and 2) proposing to use a NER data augmentation scheme, mention replacement, to alleviate the issue of unseen name style in the case of cross-domain NER. The second challenge we identify is the limited range of transformers-based models, which can be detrimental to performance in some tasks. We propose to retrieve relevant context at the document level to mitigate the lack of information induced by that lack of range, and show that it can increase performance for the NER task.

Finally, we present contributions in character network applications on two case studies. First, we leverage networks modeling different types of interactions (co-occurrence, mention and conversation) on an analysis of Alfred de Musset's *Lorenzaccio*. By using community detection on a co-occurrence network, we identify subplots, quantify their relative importance and find interactions between them. Additionally, we propose a method to detect automatically conspiracies using our dynamic mention and conversational networks. Second, we propose to leverage character networks to perform narrative matching (i.e. the task of matching corresponding narrative units) on three different adaptations of George R. R. Martin's *A Song of Ice and Fire* across media: the original novels, the comics directly adapted from these and the HBO TV show. Our results show that network-based methods can outperform existing text-based ones, and can even be combined with them to increase performance. We also highlight the importance of working on commensurate narrative units. In these two case studies, we leverage dynamic networks and show their interest, despite their relative underusage in the character network literature.

ACRONYMS

API ..... Application Programming Interface

BERT ..... Bidirectional Encoder Representations from Transformers

BIO ..... Beginning, Inside, Outside

BM25 ..... Best Match 25

CNN ..... Convolutional Neural Network

DNN ..... Deep Neural Network

E2E ..... End-To-End

GNN ..... Graph Neural Network

GPT ..... Generative Pre-trained Transformer

IE ..... Information Extraction

IO ..... Inside, Outside

LLM ..... Large Language Model

MPNN ..... Message Passing Neural Network

NER ..... Named Entity Recognition

NLP ..... Natural Language Processing

OWTO ..... Out With The Old, Dekker et al. [59]'s NER dataset

PDNC ..... Project Dialogism Novel Corpus

Renard ..... Relationships Extraction from NARrative Documents

RNN ..... Recurrent Neural Network

TF-IDF ..... Term Frequency - Inverse Document Frequency

XML ..... Extensible Markup Language

CONTENTS

1 INTRODUCTION 1
- 1.1 Contributions 3
- 1.2 Published Articles and Resources 4
- 1.3 Organization of the Manuscript 6

2 CHARACTER NETWORKS BACKGROUND 7
- 2.1 A Definition of Character Networks 8
- 2.2 Generic Extraction Framework 9
- 2.3 Topological Measures 11
- 2.4 Graph Learning 14
- 2.5 Applications 15
  - 2.5.1 Literary Analysis 15
  - 2.5.2 Classification Tasks 16
  - 2.5.3 Other Tasks 17
  - 2.5.4 Tasks on Other Media 18
- 2.6 Conclusion 18

3 A MODULAR EXTRACTION PIPELINE 19
- 3.1 Natural Language Processing Tasks 20
  - 3.1.1 Terminology 20
  - 3.1.2 Named Entity Recognition 21
  - 3.1.3 Coreference Resolution 23
  - 3.1.4 Character Unification 24
  - 3.1.5 Speaker Attribution 25
- 3.2 The Novelties Dataset 26
  - 3.2.1 Novelties v0.1.0 26
  - 3.2.2 Novelties v1.0.0 27
- 3.3 Renard: A Character Network Extraction Pipeline 29
  - 3.3.1 Motivation 29
  - 3.3.2 Design and Main Features 30
  - 3.3.3 NLP Tasks Implementation 31
- 3.4 Evaluation and Error Analysis 34
  - 3.4.1 Evaluation Measures 34
  - 3.4.2 Evaluation Corpus 35
  - 3.4.3 Pipeline Performance 35
  - 3.4.4 Comparison with Large Language Models 37
  - 3.4.5 NER Impact Analysis 39
  - 3.4.6 Coreference Resolution Impact Analysis 40
- 3.5 Conclusion 41

4 DATA AUGMENTATION FOR DETECTING NAMES OF UNSEEN STYLES 43
- 4.1 Unseen Name Styles 44
- 4.2 Cross-Domain NER and Mention Replacement 44
- 4.3 Experiments 45
  - 4.3.1 Mention Replacement 45
  - 4.3.2 Datasets 47
  - 4.3.3 Training and Evaluation 47
- 4.4 Results 47

4.4.1 Class Imbalance 49
4.4.2 Name Variety and Ambiguous Occurrences 49
4.5 Adaptation from Literary Texts to a Specific Genre 50
4.5.1 Results 51
4.6 Conclusion 51
5 CONTEXT RETRIEVAL TO ALLEVIATE TRANSFORMERS RANGE LIMITATION 53
5.1 Related Work 54
5.2 Method 55
5.2.1 Document-Level Context Retrieval 56
5.2.2 Context Retrieval Dataset Generation 56
5.3 Experiments 58
5.3.1 Unsupervised Retrieval Baselines 58
5.3.2 Inference 58
5.3.3 Re-ranker Baselines 59
5.3.4 Re-Ranker Training 59
5.3.5 Named Entity Recognition Model 60
5.3.6 Evaluation 60
5.4 Results 60
5.4.1 Neural Re-Ranker Configuration Selection 60
5.4.2 Comparison with Unsupervised Retrieval Methods 61
5.4.3 Comparison with Re-Rankers 62
5.4.4 Size of the Context Window 62
5.5 Conclusion 64
6 CHARACTER NETWORK APPLICATIONS 65
6.1 Analysis of Theater Plays: A Case Study on Lorenzaccio 65
6.1.1 Character Network Extraction 66
6.1.2 Extracting Subplots with Community Detection 68
6.1.3 Mention and Conversational Networks 74
6.1.4 Takeaways 76
6.2 Narrative Alignment: A Case Study on A Song of Ice and Fire 77
6.2.1 Corpus 78
6.2.2 General Framework 79
6.2.3 Text-Based Method 81
6.2.4 Structure-Based Method 82
6.2.5 Hybrid Method 85
6.2.6 Takeaways 87
6.3 Conclusion 89
7 CONCLUSION 90
7.1 Contributions 90
7.2 Perspectives 92
A APPENDIX 111
A.1 Chapter 3 111
A.1.1 Novelties v0.1.0: OWTO Correction Process 111
A.1.2 Large Language Models Network Extraction Prompts 113
A.2 Chapter 6 115
A.2.1 Structural Matching 115
A.2.2 Hybrid Matching 116
A.2.3 *Novels* vs. *TV Show* over U2 116

A.2.4 Cumulative Networks 118

1

INTRODUCTION

Contents

1.1 Contributions 3
1.2 Published Articles and Resources 4
1.3 Organization of the Manuscript 6

In his 2003 book, Woloch [224] chooses to perform literary analysis through the lens of *characters* rather than *plot*. For this purpose, he introduces the concept of *character-system*: each character defines its own *character-space*, its position in a space composed of other elements of the plot such as places or other characters, and the union of these character-spaces forms the character-system. This idea that a fictional narrative can be represented through its characters has found its way to Computer Science and Network Science researchers in the form of *character networks*: graphs where vertices represent characters and edges their relationships (see Figure 1 for an example). Since character networks transcribe a sort of character-system [138], they have largely been applied to literary analysis [69, 71, 100, 138, 169, 170, 182]. Researchers in that area take advantage of topological measures developed to describe complex networks, to complement traditional "close reading" approaches (i.e. the detailed manual interpretation and analysis of a text) and obtain new insights. But literary analysis is not the only domain where character networks have found success. Graphs are expressive objects that can be used to model and solve numerous problems. In the case of character networks, vertices can have individual properties and edges can reflect different kinds of relationships. The temporal evolution of relationships can also be represented through dynamic networks (as opposed to static networks that depict an entire work in a single graph). Overall, this expressiveness makes character networks very versatile. Additionally, they can be leveraged by machine learning algorithms through graph learning techniques [232]. Thus, they can be used to tackle automatically a number of tasks, such as automatic literary genre classification [23, 24, 91, 93], character classification [115, 140, 167, 169], story generation [173–175], story segmentation [136], recommendation [115], storyline detection [152, 214–216], narrative alignment [9] (see Section 6.2) or summarization [44, 45, 62, 198–200]. This profusion of applications shows that the social structure of fictional works is relevant for many problems.

While character networks are useful in a number of tasks, their manual extraction poses a resource problem as it is costly, so much so that some researchers retort to crowdsourcing [170]. This prohibitive cost creates an interest in the automatic extraction of character networks, which represents a challenge for the AI community. Depending on the medium (cinema, theater, novels...), different subfields of AI are concerned, mainly Natural Language Processing (NLP), computer vision and speech processing. In the case of novels, the main focus of this thesis, NLP in particular is leveraged to detect characters and their occurrences through the tasks of Named Entity Recognition (NER) and coreference resolution, and to understand their interactions with tasks such as sentiment analysis, relationship extraction or event detection. With the advent of pre-trained transformer-based models [60, 203] and, more recently, Large Language Models (LLMs) [46], NLP has enjoyed rapid progress in many of these tasks. NER performance has progressed from 88.76 F1 in 2003 [74] to 94.6 F1 in 2021 [208] on the widely used CoNLL-2003 dataset [195], and coreference resolution has pro-

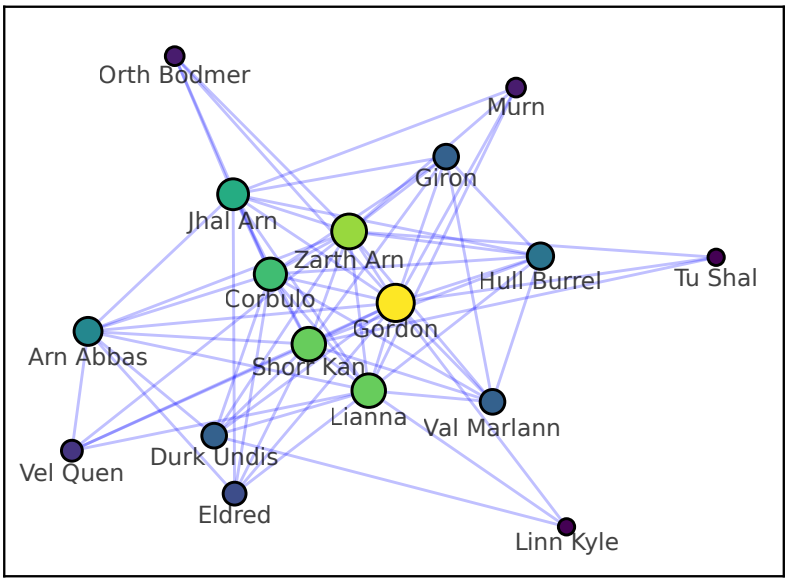


Figure 1.: Co-occurrence character network automatically extracted from Edmond Hamilton's *The Star Kings* using Renard [20]. Vertex color and size represent degree.

gressed from 64.21 CoNLL F1 in 2016 [222] to 83.3 in 2022 [40] on the popular OntoNotes v5 dataset [213]. These improvements are beneficial to character networks, as better solving NLP tasks needed for extraction is likely to result in better networks.

However, progress remains limited in some aspects. Some tasks, such as coreference resolution, are still difficult and far from solved [118]. Moreover, processing long texts (such as full novels) is challenging and resource intensive [192]. This is particularly the case for transformer-based models: while they are the backbones of recent advances in NLP, the quadratic complexity of their attention mechanism in the number of input tokens is a weakness when it comes to processing long sequences. A common workaround is to process long documents in chunks, with a rolling window, which can have adverse consequences on performance depending on the task at hand [16, 17]. Some research directions are explored to alleviate that problem: sparse transformers, such as Longformer [37] or Big Bird [237], are tailored to process long texts due to their lower complexity compared to the vanilla transformer. Still, depending on their length, long sequences can remain difficult [192]. Another direction is information retrieval: instead of taking a whole document as input, it is possible to retrieve relevant contextual information in order to handle the task locally. This information can either be external, i.e. it comes from a separate knowledge base, or internal, from the same document. For NER, since most existing datasets consists of short documents that can be processed in a single pass, context retrieval is most often external [120, 123, 180, 209, 240]. As external sources are not always available, this motivates us to propose a document-level retrieval system in this thesis.

The difficulty to work with long texts is also a motivation for the usage of character networks, as they can be used to represent them with a simple and lightweight mathematical object: a graph. Since it is difficult to obtain a text-based embedding of a long document, as highlighted above, character networks are a good candidate to fulfill that representational need, as evidenced by the many applications they are useful for. Still, recent graph learning techniques such as graph neural networks (GNNs) remain underexplored for some applications (see Chapter 2 for more details), which prompts for more research in that area.

An additional limit of recent progress in NLP is the lack of literary datasets. This is an important issue when training or evaluating models, as the lack of data leads to lower performance. Some datasets exist for singular tasks, such as OWTO [59] for NER and PDNC [205] for speaker attribution. But, arguably, one of the most central literary dataset is Litbank. It has annotations for NER [30], coreference resolution [29], event detection [183] and speaker attribution [182]. Unfortunately, Litbank excerpts are short, each having a length of approximately 2,000 words, which prevents from studying and improving the performance of models on long documents of the size of a novel. Overall, the lack of literary data is a problem, and we tackle it in two ways in this thesis. First, by proposing a new literary NER dataset, *Novelties* [10, 11]. Second, by trying to mitigate the effects of this lack of data on NER performance using data augmentation to solve the specific problem of unseen name styles in the training dataset.

This thesis is dedicated to character networks, their automatic extraction and their applications, with a particular focus on fictional novels. Our goals are to assess the performance of currently existing character network extraction algorithms, to contribute to their advancements and to study their applications in new contexts. Additionally, we make an effort to consider dynamic networks, which are overall less studied than static ones but can model the temporal evolution of characters' relationships. Toward these goals, we develop multiple software tools that we release freely for the community, such as our modular Renard network extraction library [20]. We also release a new literary NER dataset, *Novelties* [10, 11].

1.1 CONTRIBUTIONS

In this thesis, we make the following contributions in the field of NLP in general, and more specifically character network extraction and analysis:

- **Renard and Network Extraction Performance** In Chapter 3, we present Renard [20], our modular character network extraction pipeline. Importantly, Renard can extract static as well as dynamic networks. We propose new network quality measures and benchmark Renard by adapting the existing Litbank [29, 30, 183] dataset.
- **Network Extraction Error Analysis** Using Renard, we study the impact of NLP tasks' errors on the quality of extracted networks. We introduce a new degradation-based method to study that impact, and apply it to derive new insights. We demonstrate that NER and coreference resolution performance are crucial to network quality, but that not all coreference resolution errors have the same impact.
- **NER Dataset** As there is a general lack of existing literary NER dataset, we introduce the *Novelties* dataset in Chapter 3. Specifically, we use *Novelties v0.1.0* throughout experiments of this thesis, which is composed of approximately the first chapters of 40 novels.
- **NER Performance** In Chapters 4 and 5, we propose methods to improve NER performance for the specific case of novels, in order to improve the quality of extracted networks:
  - We apply mention replacement, a data augmentation method, to cross-domain NER in the specific setup where some name styles are not present in the training dataset. We show that it can bring important Recall improvements.
  - We highlight the importance of document-level context in the case of literary NER, and develop a method to train a neural context retriever on synthetic dataset we

generate with an LLM. We demonstrate that retrieving context using this model can improve NER performance on a dataset of novels.

- **Applications** In Chapter 6, we make contributions to character networks applications through two case studies. First, we perform literary analysis on Alfred de Musset's *Lorenzaccio* using three networks modeling different types of interactions: co-occurrence, conversation and mention. We extract subplots through community detection, and propose first steps to automatically detect conspiracies with character networks. Second, we utilize networks for narrative matching, the task of aligning narrative units of two adaptations of the same story, on different adaptation of George R. R. Martin *A Song of Ice and Fire* across media. We show that they can outperform text-based methods. To our knowledge, it is the first time that character networks are applied to narrative matching. We also highlight that they can successfully be combined with text-based methods to improve performance further. In both case study, we make use of dynamic networks.

For all of our contributions, we systematically make available the related source code, data and artifacts to encourage reproducibility.

1.2 Published Articles and Resources

Throughout this thesis, we published the following articles related to the current manuscript:

International Conferences/Workshops with Peer Review

- A. Amalvy et al. "Data Augmentation for Robust Character Detection in Fantasy Novels." In: *Workshop on Computational Methods in the Humanities (COMHUM)*. 2022, p. 03617722. URL: https://hal.archives-ouvertes.fr/hal-03617722 [12] (Chapter 4)

- A. Amalvy et al. "The Role of Global and Local Context in Named Entity Recognition." In: *61st Annual Meeting of the Association for Computational Linguistics*. Vol. 2. 2023, pp. 714–722. DOI: 10.18653/v1/2023.acl-short.62 [17] (Chapter 5)

- A. Amalvy et al. "Learning to Rank Context for Named Entity Recognition Using a Synthetic Dataset." In: *Conference on Empirical Methods in Natural Language Processing*. 2023, pp. 10372–10382. DOI: 10.18653/v1/2023.emnlp-main.642 [16] (Chapter 5)

- A. Amalvy et al. "The Role of Natural Language Processing Tasks in Automatic Literary Character Network Construction." In: *31st International Conference on Computational Linguistics*. 2024 [21] (Chapter 3)

International Peer-Reviewed Journals

- A. Amalvy et al. "Interconnected Kingdoms: Comparing 'A Song of Ice and Fire' Adaptations Across Media Using Complex Networks." In: *Social Network Analysis and Mining* (2024). DOI: 10.1007/s13278-024-01365-z [9] (Chapter 6)

- A. Amalvy et al. "Renard: A Modular Pipeline for Extracting Character Networks from Narrative Texts." In: *Journal of Open Source Software* 9.98 (2024), p. 6574. DOI: 10.21105/joss.06574 [20] (Chapter 3)

NATIONAL CONFERENCE WITH PEER REVIEW

- A. Amalvy et al. "Remplacement de mentions pour l'adaptation d'un corpus de reconnaissance d'entités nommées à un domaine cible (Mention Replacement for the Adaptation of a Named Entity Recognition Dataset to a Target Domain)." In: *29e Conférence sur le Traitement Automatique des Langues Naturelles*. Vol. 1. 2022, pp. 197–204. URL: http://talnarchives.atala.org/TALN/TALN-2022/8782.pdf [14] (Chapter 4)

- A. Amalvy et al. "Apprendre à classer le contexte pour la reconnaissance d'entités nommées en utilisant un jeu de données synthétique (Learning to Rank Context for Named Entity Recognition Using a Synthetic Dataset)." In: *COnférence en Recherche d'InformAtion*. 2024. URL: http://coria.asso-aria.org/2024/articles/abstract_10/main.pdf [19] (Chapter 5)

ORAL COMMUNICATIONS

- A. Amalvy et al. "Dynamique des relations dans Lorenzaccio : modélisation par réseaux complexes (Relationships Dynamics in Lorenzaccio: Modeling through Complex Networks)." In: *Journées d'Informatique Théâtrale*. 2024. URL: https://ictt.univ-avignon.fr/jit2024en/ [8] (Chapter 6)

SOFTWARES

- Renard[1] (Relationships Extraction from NARrative Documents): a modular character network extraction pipeline in Python.

- Tibert[2]: A Python package for end-to-end coreference resolution with a friendly API.

- Grimbert[3]: A Python package for neural speaker attribution in novels.

DATASETS

- Novelties v0.1.0 and v1.0.0 [10, 11] (Chapter 3): A literary NER and character unification dataset.

MODELS

- `compnet-renard/bert-base-cased-literary-coref`[4]: An English literary coreference model trained on Litbank [29].

- `compnet-renard/bert-base-cased-literary-NER`[5]: An English literary NER model trained on Novelties v0.1.0 (see Chapter 3).

- `compnet-renard/camembert-base-literary-NER`[6]: A French literary NER model trained on the dataset described by Alrahabi et al. [6].

- `compnet-renard/bert-base-cased-NER-reranker`[7]: A model trained on synthetic data to retrieve useful context to assist NER models (see Chapter 5).

1 https://github.com/CompNet/Renard
2 https://github.com/CompNet/Tibert
3 https://github.com/CompNet/Grimbert
4 https://huggingface.co/compnet-renard/bert-base-cased-literary-coref
5 https://huggingface.co/compnet-renard/bert-base-cased-literary-NER
6 https://huggingface.co/compnet-renard/camembert-base-literary-NER
7 https://huggingface.co/compnet-renard/bert-base-cased-NER-reranker

- `compnet-renard/spanbert-base-cased-literary-speaker-attribution`[8]: A literary speaker attribution model trained on PDNC [205] (see Chapter 3)

1.3 Organization of the manuscript

This thesis is organized as follows. In Chapter 2, we formally introduce the notion of character networks, provide an overview on how to extract them from written narratives, and of their multiple applications. In Chapter 3, we delve into the details of the extraction process. We survey the NLP tasks needed to extract networks, and present Renard, our extraction pipeline. We describe this pipeline in detail, assess its performance and study the impact of errors on the quality of extracted networks. Then, in Chapter 4 and 5, we strive to alleviate some of these errors. In Chapter 4, we tackle the problem of cross-domain NER (specifically, the case of unseen entity names in the training dataset) using mention replacement, a data augmentation technique. In Chapter 5, we focus on the range limitation issue of transformer-based models and its impact on NER performance, and mitigate it by training a model to retrieve relevant context at the document level. We dedicate Chapter 6 to contributions highlighting the applications of character networks. First, we show how they can be used for literary analysis in a case study on *Lorenzaccio*, a 19th century French theater play by Alfred de Musset. Second, we demonstrate how they can be used for narrative matching in another case study on George R. R. Martin's *A Song of Ice and Fire*, a contemporary series of fantasy novels that was adapted to multiple media. In both cases, we employ dynamic networks, highlighting their interest despite the fact that they are underused compared to static ones. Finally, we summarize our work and propose some perspectives in the closing Chapter 7.

8 https://huggingface.co/compnet-renard/spanbert-base-cased-literary-speaker-attribution

2

CHARACTER NETWORKS BACKGROUND

Contents

2.1 A Definition of Character Networks 8
2.2 Generic Extraction Framework 9
2.3 Topological Measures 11
2.4 Graph Learning 14
2.5 Applications 15
2.5.1 Literary Analysis 15
2.5.2 Classification Tasks 16
2.5.3 Other Tasks 17
2.5.4 Tasks on Other Media 18
2.6 Conclusion 18

The abundant literature concerning character networks is a testament to their usefulness. Their ability to serve as a compact representation of a narrative makes them extremely versatile. Networks can be used as-is, as a support for narrative visualization or descriptive analysis, or as input to graph learning algorithms to solve many different tasks such as classification, recommendation or story segmentation. Furthermore, they can be extracted from different types of media (novels, theater plays, movies, comics...), and can therefore act as a unified representation between these different media. This unified representation property has interesting applications, for example the ability to use character networks extracted from different media in the same task. We show in Section 6.2 how that property can be useful in a concrete case. More generally, character networks are an example of *complex networks*: networks that have specific properties often found when they describe real-world systems. Complex networks are used in a lot of research areas [54], from neuroscience [35] to electric power system design [176]. This means that character network applications can benefit from tools developed for other purposes and vice versa.

The interest of character networks naturally poses the question of how to extract them, and of the efforts required to do so. As stated earlier, they can be extracted from many types of media. Depending on the medium, doing so manually can be very time-consuming. In the case of novels, the main focus of this thesis, one has to annotate which characters appear and when and how they interact together. This process is very costly, so much so that some authors rely on simplified annotation frameworks: for example, Rochat and Triclot [170] retort to a simple page-level index of character occurrences. Researchers therefore developed AI methods to automatically extract character networks, helped recently by the rise of deep learning techniques and their application to NLP. For novels specifically, the recent advances in information extraction tasks such as NER or coreference resolution are particularly impactful.

In this chapter, we present the general field of character networks, restricting ourselves to novels unless stated explicitly. That chapter is meant as an overview rather than an exhaustive survey of the field. We start by giving a definition of character networks along with examples in Section 2.1. Then, we present the generic extraction framework from Labatut and Bost [110] in Section 2.2. It summarizes previous work on automatic extraction in a single, generic

pipeline. We only give a high-level overview of that pipeline: we dedicate Chapter 3 to the NLP tasks that constitute it. We review the main topological measures that are important in the context of character network applications in Section 2.3. Before focusing on these various applications in Section 2.5, we propose a brief overview of graph learning techniques in Section 2.4.

2.1 A DEFINITION OF CHARACTER NETWORKS

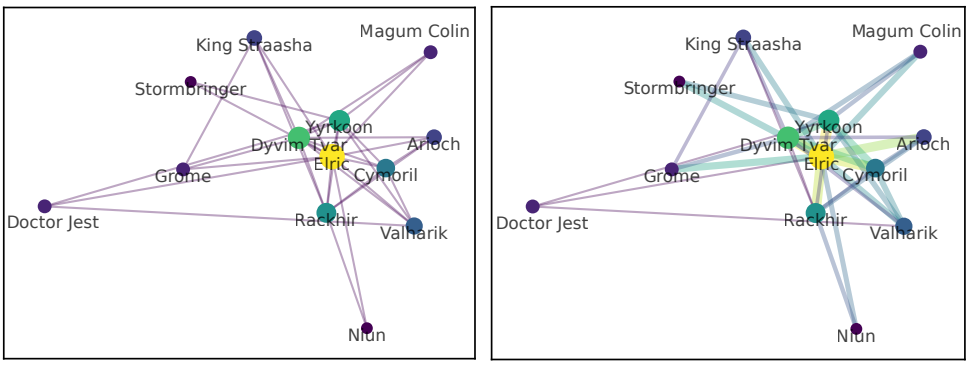
Before giving a definition of character networks, let us start by quickly defining what is a graph[1]. A *simple graph* is a pair $G = (V, E)$, where V is a set of vertices and $E \subseteq V^2$ a set of edges between them. It can alternatively be represented with its *adjacency matrix* A_G, where cell $A_{G_{ij}}$ of the matrix is 1 if vertices i and j are connected by an edge, or 0 otherwise. A simple graph can model the relationships between a set of related elements, but only in a binary way: either there exists an edge in E between two vertices of V or not. To model more complex relationships, other and more general types of graphs can be employed. *Directed graphs* have oriented edges, meaning the pairs of vertices that form them are ordered, which enables them to model asymmetric relationships. *Weighted graphs* have weighted edges, which can be represented as triplets $(u, v, w) \in V \times V \times \mathbb{R}$ where w is the weight of the edge, to model the intensity of relationships. *Signed graphs* have signed edges $(u, v, s) \in V \times V \times \{-, +\}$, where the sign s can model the polarity of a relationship. For all of these different types of graph, the adjacency matrix is slightly adapted. For directed graphs, a non-zero element $A_{G_{ij}}$ indicates an edge from i to j. For weighted graphs, elements of the matrix are not restricted to 0 or 1 but can take a value in $\mathbb{R}$ denoting edge weight. For signed graphs, elements can take a value in $\{0, 1, -1\}$.

Simply put, a character network is a graph (or a series of graphs in the case of dynamic networks) extracted from a narrative medium where vertices V represent characters while edges E represent their relationships. This definition, while simple, does not reveal the diversity of networks that it is possible to extract, as can be seen in Figure 2. Vertices can have attributes that make them more informative, such as the social category of a character, its gender, its wealth or its allegiance. As defined earlier, edges can be of different types. An unweighted edge only represents a binary status of relationship (see Figure 2a), but a weighted edge often represents the importance of a relationship between two characters (Figure 2b). A directed edge can depict asymmetric relationships (Figure 2d), while a signed edge can represent the polarity of their relationship (Figure 2c).

Since networks are extracted by different means, the meaning of vertices and edges directly depends on the underlying extraction algorithm. By definition, vertices always represent characters, but edges can represent a variety of different relationships: co-occurrences between characters in a text, conversations between them, direct interactions or even precise social relationships.

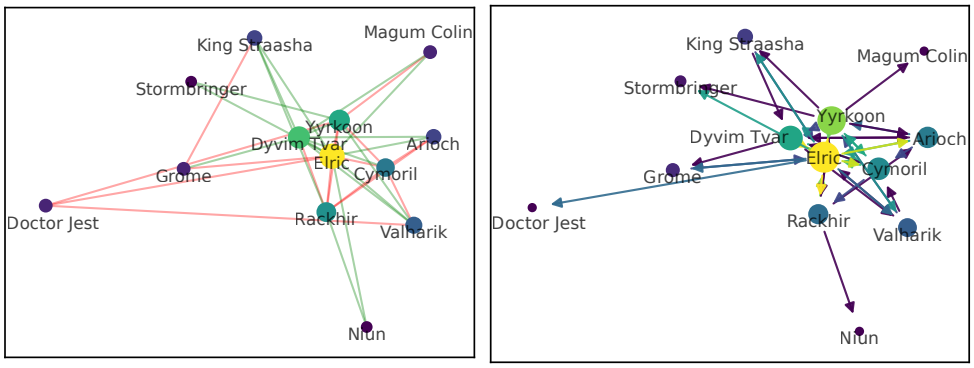
So far, we described *static* character networks: a single graph is used to represent a whole narrative. It is also possible to extract a *dynamic* character network [4, 94], which is a network that evolves through time to represent the evolution of characters and their relationships. For simplicity, such a network can be represented with a series of graphs $G = (G_1, G_2, \ldots, G_n)$ where $G_t = (V_t, E_t)$. Each graph in that series is extracted within a sliding *temporal window* that defines the temporal start and end of the represented period. These temporal windows are not necessarily disjointed, and can *overlap*. In the case of novels, a frequently used temporal window is the chapter [2, 23, 83, 160], but it can be selected depending on the appli-

1 We use the terms "graph" and "network" interchangeably throughout this manuscript.



(a) Simple network, where two characters are linked if they co-occur in the text.

(b) Weighted network. Edge width and color indicate the number of co-occurrences.



(c) Signed co-occurrence network obtained with additional sentiment analysis.

(d) Directed mention network. The direction of the arrow indicates the speaker and mentioned character. Edge width corresponds to the number of mentions.

Figure 2.: Different character networks from Moorcock's *Elric of Melniboné*, automatically extracted using Renard [20]. In all of these networks, vertex size and color represent their degree.

cation. When each graph of the series represents only the interactions and characters from its assigned temporal period, the dynamic network is said to be *instant*. By contrast, in a *cumulative* network, each graph consists of all the characters and interactions that occurred in and before the considered period. Thus, the final cumulative network is equal to the static network extracted from the same source. Figure 3 illustrates the extraction of a dynamic network with overlapping temporal windows.

2.2 GENERIC EXTRACTION FRAMEWORK

Now that we discussed the basics of character networks, we focus on how to extract them. In their survey, Labatut and Bost [110] explore the works of different researchers interested in character networks in fiction. From these works, they derive a generic 3-phase extraction framework that can be seen in Figure 4. This framework is generic relative to the source medium (novels, films, ...), but for our purposes we limit ourselves to novels.

In the first phase, extraction systems strive to extract characters in a given text. They start by detecting character occurrences in the source text. Depending on the type of extracted network, these occurrences can comprise character appearances, evocations, or both. Then, they *unify* these occurrences by associating each of these to a unique character. This process amounts to detecting characters and all of their mentions in the source text.

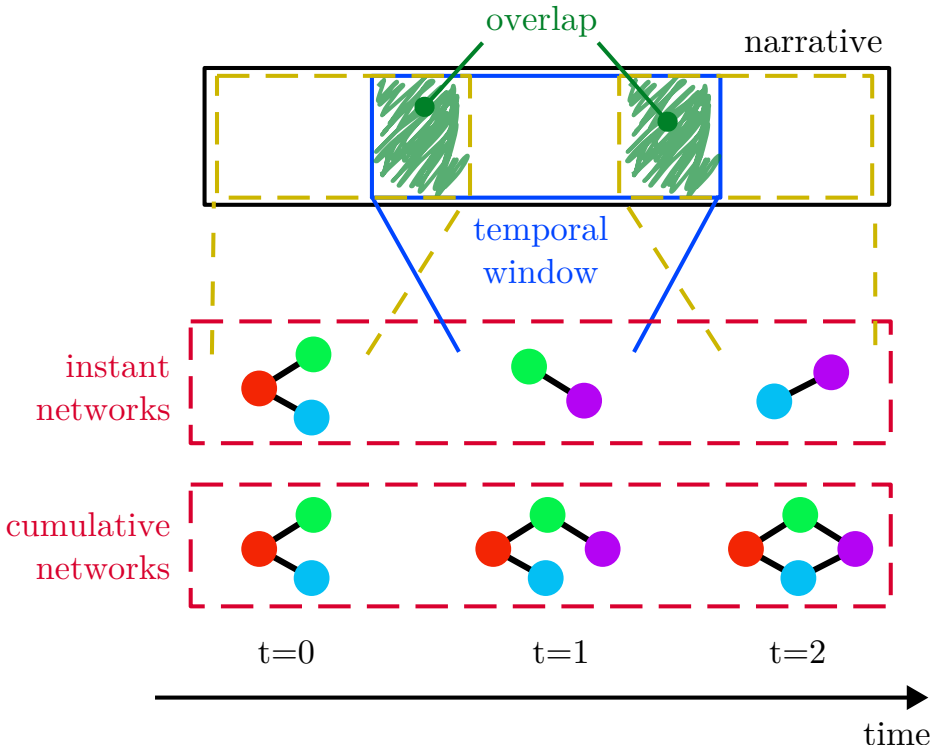


Figure 3.: Example of a dynamic character network extracted from a narrative. In this example, the narrative is cut in three temporal windows with overlap.

The second phase consists of detecting interactions between characters. Depending on the application, one can extract different kinds of interactions: Labatut and Bost [110] mention five different types. The simplest form is *co-occurrence*, where one considers that two characters are interacting when they are mentioned close to each other in the source text, in a window we call the *co-occurrence window*. This window can be defined with different units depending on the application: tokens, sentences, paragraphs, chapters... Co-occurrence is, of course, a less-than-perfect approximation where we suppose that two characters likely have some form of interaction when they are mentioned together in the source text, but the exact nature of that interaction is not extracted. At the opposite end of the extraction difficulty spectrum, a *direct action* interaction is recorded when a character affects another directly, for example in a physical interaction. There also exists *conversational* interactions, where we register an interaction when two characters discuss together. In conversations, one can also focus on cases where a character talks about another by focusing on *mentions* of other characters. Finally, *affiliation* interactions are meant to register the social relationships between characters. These can consist of familial (e.g. a character being the son of another one) or societal (e.g. a character being the boss of another one) relationships.

The final extraction phase consists of constructing a character network (either static or dynamic) from the previously extracted interactions. Given these, this process is purely algorithmic and completely deterministic.

The first and second phase correspond to solving NLP tasks, while the third phase is a comparatively simpler algorithmic problem. The first phase is associated to *NER*, *coreference resolution*, and *character unification* (also known as *alias resolution*). The tasks that must be

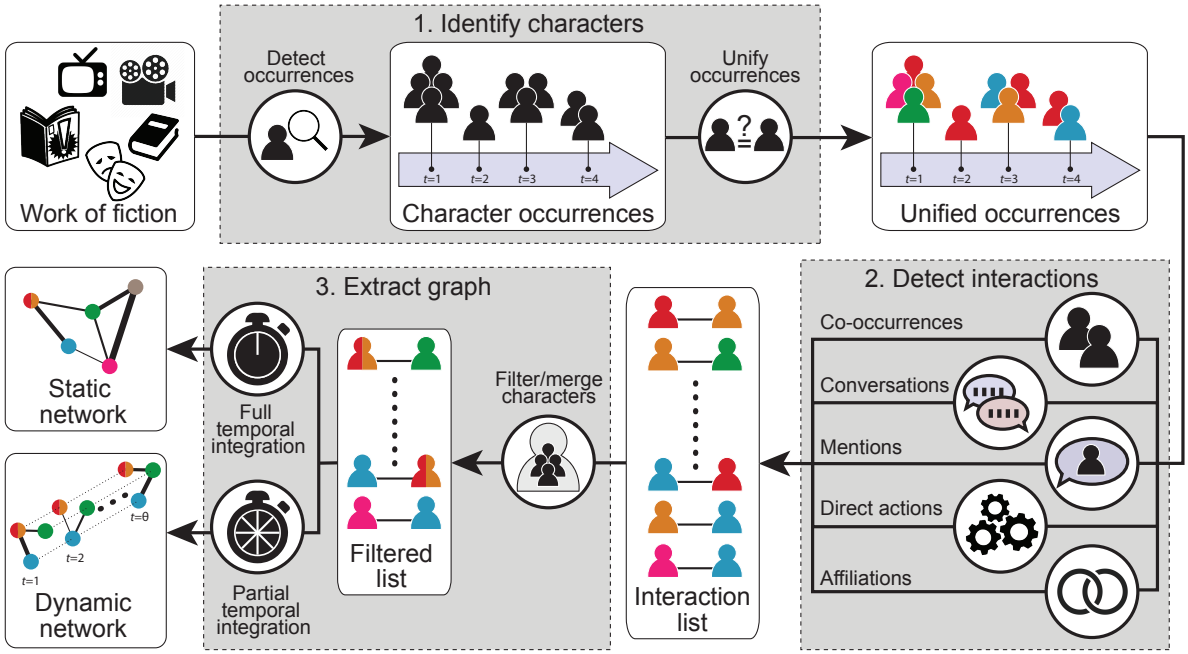


Figure 4.: The generic extraction framework from Labatut and Bost [110].

solved to carry the second phase are as diverse as the different kinds of interactions one can extract: *speaker attribution* is essential to extract conversational interactions, while *relationship extraction* can be used to extract affiliations and *event extraction* to extract direct actions. We discuss some of these tasks in more details in Chapter 3.

2.3 TOPOLOGICAL MEASURES

Once networks are extracted, information can be obtained from them. Topological measures designed for complex networks in general are commonly used for that purpose. These measures can sometimes be used as features for learning algorithms, and some of these are directly interpretable and can be used to manually extract knowledge from character networks. These measures can be defined locally, at the edge or the vertex level, or globally at the graph level. Due to their importance, we describe the most relevant ones when it comes to character networks in this section.

VERTEX CENTRALITY Centrality is a measurement of the importance of each vertex in a network, that can be interpreted as the importance of a character relative to other characters in a narrative. There are, however, different centrality measures, each with a different interpretation. Among the most used, we can cite *degree centrality*, *betweenness centrality* [22], *closeness centrality* [77], *Eigenvector centrality* [42, 43] and *vitality* [106]. In his PhD thesis, Rochat [167] discusses different network centrality measures for the purpose of character network analysis, and states their usefulness in literary analysis. Centrality has multiple uses in the context of character networks, but a common one is to determine whether a character is a main or a minor character, or even the protagonist of the story [168, 169].

To go into more details, the *degree centrality* of a vertex is its number of neighbors. At the graph level, the average degree is often used to compare different networks [69, 100, 127]. In the case of weighted networks, we can also define the *strength* of a vertex, which is a

generalization of the concept of degree. It is computed as the sum of the weights of the edges connected to the vertex of interest.

Meanwhile, the *betweenness centrality* corresponds to the proportion of shortest paths that pass through a vertex. It is computed as follows:

$$c_B(v) = \sum_{w,x \in V \setminus \{v\}} \frac{\sigma(w,x|v)}{\sigma(w,x)}, \tag{1}$$

where $\sigma(w,x)$ is the number of shortest paths between vertices w and x, and $\sigma(w,x|v)$ the number of these paths passing through vertex v. The betweenness centrality can be interpreted as the control of a vertex on flows in the network [76].

The *closeness centrality* [77] of a vertex gives information on its *distance* to all other vertices, the distance between two vertices being the length of the shortest path between them. It is the inverse of the average distance between a given vertex and all the other:

$$c_C(v) = \frac{|V| - 1}{\sum_{w \in V \setminus \{v\}} d(v,w)}, \tag{2}$$

where $d(v,w)$ is the distance between vertex v and w.

The *Eigenvector centrality* [43] of a vertex is based on the centrality values of its neighbors. The Eigenvector centrality of a vertex is higher when it is connected to vertices that are themselves of high centrality, and vice versa. Given A_G the weighted adjacency matrix of graph G, we can find the centrality vector $\mathbf{c_E}$ as the solution of the following equation:

$$\mathbf{c_E} = A_G \mathbf{c_E}. \tag{3}$$

Then, $c_E(v)$ is simply the entry of vector $\mathbf{c_E}$ that corresponds to the index of vertex v in the adjacency matrix. From that equation, one can see that the centrality of a vertex depends on the centrality of its neighbors, as each entry i of vector $\mathbf{c_E}$ is the dot product of $\mathbf{c_E}$ itself with the i-th row of the adjacency matrix A_G, that represents the neighbors of the i-th vertex.

Finally, *vitality* [106] measures the impact of *removing* a vertex from a network, and is used in literary analysis for its interpretability [138]. Vitality measures the impact of a vertex deletion using the change of a graph-level measure, and therefore depends on an auxiliary function f:

$$c_v(v,f) = f(V \setminus \{v\}) - f(V) \tag{4}$$

CENTRALITY DISTRIBUTION Authors are also interested in the *distribution* of centrality in the network, often degree centrality [108, 127, 152, 167, 185]. Since the degree distribution of many real-world social networks exhibits the *scale-free* property [33] (that is, it follows a power law), this property is used by authors to assess the realism of the networks. In a *scale-free* network, a few of the vertices are important hubs with high degree, while most vertices have low degree.

ASSORTATIVITY Assortativity is a measure of how correlated connected vertices of a graph are in terms of a given measure or attribute. While assortativity can be computed for any attribute, the most often used form is degree assortativity, that measures if high degree vertices are mostly connected to high degree vertices and vice versa. Thus, the term "assortativity" alone often refers to degree assortativity in practice. Since real-world social networks are often degree assortative, degree assortativity is sometimes used as an indication that a network is realistic or not, similarly to degree distribution [127].

COMMUNITIES As vertices can have attributes, they can be partitioned in communities, either completely disjoint or overlapping depending on the method that is employed. While some authors extract these communities manually [170], a number of automated extraction algorithms exist [75], even for the case of dynamic networks [25, 36, 190]. Community detection is used by researchers to detect subplots involving groups of characters [41, 128] (as we do in Section 23). For signed networks, a special version of the problem is called *correlation clustering* [32]: it consists of finding communities such that most positive edges lie inside communities, while negative edges lie between them. These communities are straightforward to interpret as groups of allied characters opposed to other groups.

DENSITY Graph density is the number of existing edges in the networks divided by its number of possible edges. Since the number of possible edges in a network is $\binom{|V|}{2}$, we can compute density as:

$$D(G) = \frac{|V|}{\binom{|V|}{2}}. \tag{5}$$

A denser graph means characters have more interactions between each other, while a less dense graph may be indicative of more separate social groups.

DISTANCE The distance between two vertices in a network can be interpreted as the social connection or disconnection between characters. Two graph-level measures related to distance are often used in the literature. First, the diameter of a network is the maximal distance over its pairs of vertices. Second, the average distance is simply the average distance over all pairs of vertices.

STRUCTURAL BALANCE In the case of signed networks, each edge is either positive (+) or negative (-) and often represents the polarity of the relationship between two characters (friendly or unfriendly). In the case of unweighted networks, there are four possible types of triads: `+++`, `++-`, `+--` and `---`. `+++` and `+--` triads are considered *balanced* because they are stable from a social point of view. Meanwhile, `++-` and `---` triads are considered *imbalanced* due to their social instability [48]. At the graph level, structural balance is simply the proportion of balanced triads in the network. It has a very direct interpretation: if structural balance is high, then the number of social conflicts is probably high, and vice versa.

CLUSTERING COEFFICIENT The clustering coefficient refers to two distinct metrics: the global clustering coefficient, at the graph level, and the local one, at the level of vertices. Generally, it represents the extent to which vertices in a graph are grouped together. At the graph level, the global clustering coefficient is the number of closed triads (three vertices connected together) over the number of open ones (a group of three vertices with only two connections). At the vertex level, it is the number of edges between its neighbors divided by the same quantity if all neighbors were connected. It is equivalent to the global clustering coefficient, if we limit ourselves to triads centered on a vertex.

k-CORES The k-core of a graph is its largest subgraph so that all its vertices have a degree of at least k. Some authors use it to separate the core cluster of the story and the less important characters [151, 152, 179].

One of the advantages of networks is that they can be used as inputs to many machine learning algorithms, which makes them very versatile. Since these algorithms can be used in character network applications in general, we provide a brief overview of graph-level learning in this subsection. We also want to mention *graph embedding*, a specific form of graph learning that allows to represent a graph as a vector in a meaningful vector space. This is very useful, since in the case of character networks, one can represent an entire novel in a single vector. Formally, given the set of all possible graphs $\mathcal{G}$, a graph embedding function can be defined as follows:

$$\begin{aligned} f &: x \to f(x) \\ &\mathcal{G} \mapsto \mathbb{R}^k. \end{aligned}$$

Yang et al. [232] provide an overview of existing graph learning techniques. They classify existing them into four categories:

- **Traditional Learning** Before the advent of deep learning models, there were already several techniques to learn graphs. Graph kernels leverage pairwise similarity between graphs, while pattern mining involves representing graphs using discriminative subgraphs. It is also possible to represent a graph using a set of manually defined features, such as centrality or other properties. We described some of these features in Section 2.3. Compared to methods based on deep learning, while these may not give the best results, features-based techniques are often more interpretable.

- **Graph-Level Deep Neural Networks (GL-DNNs)** With the general progress in deep learning for images and text, researchers attempted to adapt deep neural networks to graphs. For example, Graph2Vec [145] is an adaptation of the original skip-gram algorithm [135] to graphs. Other techniques based on recurrent neural networks (RNN) [67] and convolutional neural networks (CNN) [78] also exist.

- **Graph-Level Graph Neural Networks (GL-GNNs)** More recently, instead of adapting deep neural networks meant for other modalities, neural networks specialized for graphs have been developed. Among these, message passing neural networks (MPNN) [81] are particularly influential. MPNNs operate in two phases: the first phase, message passing, updates vertices representation in a graph, while the second phase, readout, computes a graph-level representation using the vertex-level representations obtained previously.

- **Graph Pooling** Since many GL-DNNs and GL-GNNs algorithms derive representations at the vertex level, graph pooling techniques are necessary to encode graph-level representations. *Global graph pooling* techniques aggregate vertex representation direction, either by direct computation (for example, with a mean pooling over vertices) or using learning mechanisms such as CNNs or attention layers. On the other hand, *hierarchical graph pooling* techniques try to take the structure of the graph into account.

Thanks to these techniques, character networks have the capacity to be used to solve many problems. In the next section, we see the problems on which they have successfully been applied.

2.5 APPLICATIONS

As we have seen, there are many techniques that make character networks a very useful tool. Concretely, they have two major advantages: first, networks are understandable by humans, visually but also through topological measures. This allows us to analyze a source text using its extracted network, and to derive new insights that would have been hard to obtain given the sole text, in a "distant reading" fashion [170]. Second, a graph is a useful representation for solving machine learning problems. While there exist embedding techniques specialized for documents, the length of these documents is often an issue. Firstly, because of computational limitations: A transformer-based model such as SentenceBERT [164] may run out of memory when processing an entire novel. Secondly, it is not clear how to obtain a useful embedding on such long documents. Compared to graphs, embeddings seem to have an advantage: they can directly be used as inputs to machine learning methods, either for training or for inference. However, as we have seen in the previous section, there exist graph learning techniques allowing character networks to be used similarly.

These two advantages of character networks, interpretability and ease of use in machine learning algorithms, mean that there are a lot of applications for character networks. In the rest of this section, we explore the main areas where they have been applied successfully. We give examples to illustrate applications, but refrain from being exhaustive in our overview of the literature. The curious reader may refer to Labatut and Bost [110] for a more complete survey.

2.5.1 *Literary Analysis*

Character networks are largely leveraged by humanities researchers to support literary analysis. Here, we talk about "literary analysis" in the broad sense of the term without limiting ourselves to works from literary scholars: that is, the process of gaining insights on a literary work and its impact. In order to do so, authors often rely on the topological measures we describe in Section 2.3. Moretti [138] is one of the precursors of such approaches, and connects Woloch's concepts of *character-space* and *character-system* [224] to character networks, highlighting how they can be proven to be a useful tool for literary analysis.

A lot of the topological measures we present in Section 2.3 are used to derive meaning from character networks. One of the most used is *centrality*: A common usage is to determine whether a character is a main or a minor character (for instance, see Rochat [168] and Rochat and Kaplan [169]). Centrality, as a proxy for importance, can also be compared to other vertex-level measurements. For example, Grayson et al. [83] study the co-occurrence network of Jane Austen's *Sense and Sensibility*. They find that the top characters in terms of centrality in the overall network are the wealthiest, which echoes existing discussions on the social exclusivity in Jane Austen's novels. Authors also study the impact of the removal of important characters (most often, the main character) on the network using vitality. While studying the conversational network from Maria Edgeworth's *The Absentee*, Falk [71] highlights the importance of the main character by removing it from the network and noticing that other characters become disconnected as a result.

The communities found in character networks can also be used for analysis purposes. Some authors leverage an automatic community detection algorithm such as Louvain [39], while others extract communities manually for their analysis. For example, Rochat and Triclot [170] analyze a corpus of science-fiction works (not limiting themselves to novels, but also including films, series, comics and even video games) through the prism of co-occurrence character networks. They first propose a classification of these networks in different groups,

and then analyze the narratives by commenting on the place of characters in the networks depending on their social category. Other authors use communities to detected different subplots in a story [41, 128].

A number of graph-level metrics are used to compare networks for various purposes. Rochat and Triclot [170] use density when trying to categorize different works of their science-fiction corpus, and notice that low density is characteristic of behind closed-doors stories such as Ridley Scott's *Alien*. At the opposite end of the spectrum, Glukhovsky's novel *Metro 2033* exhibits a very low density due to its disconnected social communities. Relatedly, some researchers are interested in determining the realism of a story [103, 104, 126–129, 235, 236]. To do so, they often compare graph-level properties (coefficient clustering, average degree, diameter...) of networks extracted from their stories to real-life social networks. The degree distribution is often used, in particular the *scale free* [33] property of networks is often checked since many real-world social networks exhibit that property. Assortativity is also used as an indicator of the *artificiality* of the story, as real-life social networks are often assortative [146]. Mac Carron and Kenna [127–129] and Yose et al. [235] use structural balance to claim that the mythological network they study is realistic, arguing that since these networks have a high balance, they are close to real-world social networks that also display high balance.

Different networks can be used for different purposes. Co-occurrence networks are used as an overall approximation of relationship structure, but other types of interactions can be more precise. For instance, conversational networks are leveraged to answer specific literary questions. Elson *et al.* are the first to automatically extract conversational networks from novels [68–70]. Using these, they try to verify existing literary theories on a corpus of sixty nineteenth-century novels [69]: 1. There is an inverse correlation between the amount of dialogue and the number of characters in novels 2. Characters in urban settings tend to share less dialogues than characters in rural settings. Elson et al. [69] conclude against both hypotheses, but Jayannavar et al. [100] later revisit their results and find a different conclusion. Sims and Bamman [182] use conversational networks extracted with BookNLP [31] to measure information propagation between characters. They find that information flows through characters that serve as a bridge between otherwise disjointed communities, and that female characters propagate information more often than men.

2.5.2 *Classification Tasks*

GRAPH CLASSIFICATION Graphs in general have been successfully applied to text classification [206]. Therefore, character networks seem to constitute a natural candidate for classification: nonetheless, their usage for the task remains understudied.

Intuitively, the structure of a character network is linked to the literary genre and author of a novel: for example, fantasy writer George R. R. Martin includes an enormous amount of characters in *A Song of Ice and Fire*, while Jane Austen's novels focus on a comparatively smaller set of characters. Thus, it is not surprising that the two most popular classification tasks tackled in the literature are *literary genre classification* and *author classification*.

Hettinger et al. [91] work on a corpus of 1,682 German novels. They extract features based on style, topics and co-occurrence character networks, but find that network-based features obtain worse performance on their dataset than the other sets of features. Holanda et al. [93] tries to differentiate between biographical, legendary and fictional texts. They focus on a small set of 12 novels and perform a case study rather than an evaluation on a large benchmark. They use descriptive features of the networks, and conclude that these features are not very discriminative. Condello et al. [53] classify 1,800 British and American novels of the 19th century based on the author's gender and nationality. However, they only use total

character count and total link weight as features, limiting the scope of their results. Finally, Ardanuy *et al.* [23, 24] perform classification of novels based both on genre and authors. They use traditional graph description features (density, diameter, radius...) and obtain decent results on author classification (from 60 to 75 F1), but not on genre classification (from 27 to 40 F1).

In all of these works, authors use traditional graph learning techniques. Recent graph learning methods such as graph neural networks have not been applied to character networks extracted from novels, and thus constitute a promising object of study given their performance.

VERTEX CLASSIFICATION Rather than classify the whole graph, it is also possible to classify each vertex. A lot of stories present a binary division of their characters: a group assists the protagonist, while a second group forms its opposition. Therefore, Muhuri et al. [140] try to automatically find the protagonist and antagonist in networks extracted from novels. They also automatically extract communities in their networks, and find three categories of characters: a group centered around the protagonist, a group of antagonists, and a neutral group. Several authors try to distinguish between several categories of importance for characters. The simplest case assumes two classes, *main characters* and *minor characters* [115], but some authors also solve a three-class problem with *main characters*, *supporting characters* and *extras* [167, 169].

As graph classification, vertex classification on networks extracted from novels is understudied. Authors use simple vertex features (degree and other centrality measures, transitivity...), but to our knowledge no work exploits recent deep learning techniques.

2.5.3 *Other Tasks*

STORY SEGMENTATION Story segmentation is the task of decomposing the plot into sequential phases. In novels, this can for example mean splitting the story into chapters (when they are not indicated) or scenes. While there are works on this front leveraging NLP techniques, to our knowledge there is a single work using character networks for that purpose. Min and Park [136] model a narrative in terms of dynamic cumulative networks. They detect plot phases (exposition, resolution...) through the variation of vertices and edges.

STORY GENERATION Story generation is the task of automatically generating narratives. Recent methods mostly leverage LLMs to generate stories [154, 207, 227]. We are only aware of one series of work leveraging character networks: Sack [173–175] use them to generate what they call *"proto-narratives"*, which are series of events that can be used to create a story afterward. To obtain these proto-narratives, the authors start from an unbalanced network (a signed network with ++- or --- triads) and progressively balance it by switching the sign of some relationships. These switches of signs each indicate an event: either a character befriending a previously hostile character (+ to -), or a betrayal of a character by another (- to +). As the final network is balanced, social relationships have reached some sort of stability and a sense of cluster is obtained.

NARRATIVE ALIGNMENT Narrative alignment is the task of aligning narrative units (e.g. novels chapter, TV show episodes...) between two medias. It is a useful task to study the adaptation of a story from one media to another. While it is possible to carry this task using only text as input [155, 156], we show in Section 6.2 that using character networks can yield better performance, and can even be combined with text to obtain better results.

2.5.4 *Tasks on Other Media*

In this section, we generally limited ourselves to tasks for which character networks extracted from *novels* have been applied. In this subsection, we quickly list a few additional tasks where character networks extracted from *other types of media* (movies, comics...) have been applied. Since character networks can represent different types of media in a unified manner, tasks presented here may be transcribed in the context of novels.

RECOMMENDATION Character networks can be used as part of a recommendation system, as what a user likes in a work of fiction may be linked to the relationship dynamics between characters. Lee and Jung [115] implement a movie recommendation system using signed dynamic co-occurrence networks. While they perform their experiments on movies, their framework is generic enough to be used in other media.

STORYLINE DETECTION Stories are often decomposed into multiple subplots. With the assumption that each subplot is centered on a leading character, several authors attempt to perform storyline detection in movies and TV shows [153, 214–216]. They consider that storylines are a series of ordered scenes, and they assign each scene to a subplot community depending on the involved characters.

SUMMARIZATION Character networks have been applied to summarize movies and TV shows, but to our knowledge not novels. Do et al. [62] and Tran et al. [198, 199] extract networks from movies to identify the main characters in movies, and use these main characters to identify important scenes, that are used to generate a summary. Bost [44] and Bost et al. [45] use dynamic character networks to generate character-centric extractive summaries of TV shows. Finally, Tsai et al. [200] extract a network where vertices are groups of characters, identify important groups in the network and select important scenes based on the presence of these important groups.

2.6 CONCLUSION

In this chapter, we presented character networks and their applications. We started with a definition, along with a few examples. Then, we presented the generic extraction framework originally highlighted by Labatut and Bost [110]. After that, we reviewed the most relevant topological measures for character networks. These measures are often used in applications, and we also make use of them in the rest of this thesis, notably in Chapter 6. Subsequently, we provided a quick overview of graph learning techniques, in order to argue for the potential of character networks. Finally, we went over some applications of character networks.

In the next chapter, we go into more details concerning the NLP tasks needed to apply the generic extraction framework we presented in Section 2.2. These tasks are still challenging and not fully solved, and we review their impact on network extraction as part of the next chapter. Since obtaining better performance for these tasks increases the quality of the extracted networks, we dedicate a part of this thesis (Chapters 4 and 5) to reducing errors for a fundamental task: NER.

3

A MODULAR EXTRACTION PIPELINE

Contents

3.1 Natural Language Processing Tasks **20**
- 3.1.1 Terminology 20
- 3.1.2 Named Entity Recognition 21
- 3.1.3 Coreference Resolution 23
- 3.1.4 Character Unification 24
- 3.1.5 Speaker Attribution 25

3.2 The Novelties Dataset **26**
- 3.2.1 Novelties v0.1.0 26
- 3.2.2 Novelties v1.0.0 27

3.3 Renard: A Character Network Extraction Pipeline **29**
- 3.3.1 Motivation 29
- 3.3.2 Design and Main Features 30
- 3.3.3 NLP Tasks Implementation 31

3.4 Evaluation and Error Analysis **34**
- 3.4.1 Evaluation Measures 34
- 3.4.2 Evaluation Corpus 35
- 3.4.3 Pipeline Performance 35
- 3.4.4 Comparison with Large Language Models 37
- 3.4.5 NER Impact Analysis 39
- 3.4.6 Coreference Resolution Impact Analysis 40

3.5 Conclusion **41**

In Section 2.2 of the previous chapter, we present the generic character network extraction framework of Labatut and Bost [110], which is a synthesis of previous researchers' extraction efforts. In this chapter, we delve into the practical side of extraction in detail. As stated earlier, automatically extracting character networks requires solving multiple NLP tasks. We first go over the main ones we identify in Section 3.1: NER, coreference resolution, character unification and speaker attribution. We mention the main existing datasets for each task, especially datasets from the literary domain, since we focus on novels.

Literary resources for NLP tasks are lacking in general. While a few datasets such as Litbank exist [29, 30, 183], most are composed of short texts. This is an issue since some tasks, such as coreference resolution, are significantly harder when the length of documents increase [86, 87]. Thus, we propose a new literary dataset in Section 3.2: *Novelties*. It consists of annotations for the NER and character unification tasks. While our first version of Novelties (Novelties v0.1.0) consists of short documents of approximately one chapter, its second version contains fully annotated novels. We use Novelties throughout this thesis, notably to evaluate the impact of our contributions to NER in Chapter 4 and Chapter 5.

Then, we present existing character network extraction systems in Section 3.3.1. We note that these systems have drawbacks: none of them support extracting dynamic networks, they generally focus on a single type of network and some of them are not modular. This leads

us to propose *Renard* in Section 3.3, a modular character network extraction pipeline. We implement Renard in Python and make it available under a free license[1]. We provide a performance assessment of Renard in Section 3.4. To do so, we contribute new measures to evaluate the quality of an extracted network: Vertex Precision and Recall, Edge Precision and Recall, and Weighted Edge Precision and Recall.

Renard, as other existing extraction systems, is in the form of a cascading pipeline: a sequential series of steps where the output of one step feeds into the input of the next. Such systems are interpretable, but they can be subject to cascading errors. Knowing which errors have the most impact on the quality of the network is important, as it will allow prioritizing research efforts. Thus, we systematically analyze the impact of NER and coreference resolution errors in Renard, respectively in Section 3.4.5 and Section 3.4.6.

To summarize, this chapter makes the following contributions:

- We propose the Novelties literary dataset in Section 3.2, with annotations for the NER and character unification tasks.
- We propose the modular Renard character network extraction pipeline in Section 3.3. The content of this section is based on our work published in the *Journal of Open Source Software* [20].
- In Section 3.4, we evaluate Renard, compare its performance with non-cascading systems and analyze the impact of NER and coreference resolution errors on the quality of extracted networks to guide future research. This section is based on an article accepted at the COLING 2025 conference [21]. We provide the source code and data to reproduce our experiments under a free license[2].

3.1 Natural Language Processing Tasks

3.1.1 *Terminology*

Before presenting the NLP tasks relevant to character network extraction, we must first clarify the vocabulary surrounding them. Each of these tasks has a specific established terminology, but these are not always compatible together, which can cause confusion. Therefore, in this section, we define the terms we use for the entirety of this thesis. When presenting tasks below, we still mention the terms typically used in the literature as anchor points.

We first define our usage of the term *entity*:

Definition

An **entity** is a unique element having an existence inside the narrative universe.

We identify different types of entities. In this thesis, we are mainly interested in *characters*. According to the *Living Handbook of Narratology*[3], a character is a *"text- or media-based figure in a story world, usually human or human-like."*. We go further and define that:

Definition

A **character** is a sentient figure with some form of agency in a narrative.

1 https://github.com/CompNet/Renard
2 https://github.com/CompNet/Splice
3 https://www-archiv.fdm.uni-hamburg.de/lhn/

This definition includes traditional human characters, but also aliens, sentient robots or creatures, and even conscious objects such as the enchanted sword *Stormbringer* in Moorcock's *Cycle of Elric*. Other types of entities are also of interest, such as locations, organizations...

Since stories talk about entities, their relationships and their actions, they are mentioned throughout narratives. We define the *mention* of an entity as follows:

Definition

A **mention** is an occurrence of an entity in a story.

For example, in a novel, the act of mentioning a character by one of their names is a mention. This leads us to the concept of *form*:

Definition

The **form** of a mention is its textual content itself, detached from its position in the text.

Based on form, we distinguish two types of mentions, *alias mentions* and *generic mentions*:

Definition

An **alias mention** is a mention whose form can strongly identify an entity. It must be self-sufficient, identify the entity uniquely, and its form must appear frequently.

We directly derive this definition of *alias mention* from the guidelines of Novelties [10], our NER dataset that we present in Section 3.2. The form of such mentions is said to be an *alias*, which can be a proper noun, like the name of a character, or a definite description (i.e. an expression of the form *determiner + noun phrase* such as *the President of the United States*). Finally:

Definition

A **generic mention** is any mention that is not an alias mention.

A *generic mention* can be a pronoun, or a generic expression such as *a man*, *a little girl*... But, by itself, is not sufficient to identify a character.

3.1.2 *Named Entity Recognition*

NER is a fundamental task in information extraction (IE). It started getting attention thanks to the MUC conferences[4], that took place from 1987 to 1998. NER consists of two subtasks: detecting a subset of entity mentions in texts[5], and classifying the type of the underlying entity given a set of predefined classes [66]. Different corpora observe different rules regarding the mentions to detect, but in our case we are mainly interested in alias mentions. The classes of entities to detect also depend on the corpus. One of the most popular set of entities comes from the CoNLL-2003 shared task [195], and contains four classes: persons (`PER`), organizations (`ORG`), locations (`LOC`) and miscellaneous (`MISC`).

TASK FORMALIZATION NER is often formalized as a sequence-labeling task [230]. Tokens that are not part of a mention are part of the `O` ("outside") class, while tokens that are part of a mention are classified depending on the underlying entity class and the position of that token. There are several ways of tagging in-mention tokens, the most simple scheme being `IO`:

4 https://dl.acm.org/conference/muc

5 In the usual NER terminology, the task is about detecting "entities". However, in this thesis, we use the term "entity" to refer to *the concept being mentioned*, not one of its mentions.

Example from Edmond Hamilton's *The Star Kings*

I-PER	I-PER	O	O	O	I-PER	I-PER	O	I-PER	O	O	O	I-LOC	O	O	0
Shorr	Kan	said	to	bring	Zarth	Arn	and	Lianna	back	to	the	Cloud	at	once	.

In the IO scheme, tokens that are *inside* an entity are tagged `I-TYPE`, where type denotes the type of the underlying entity. While the IO scheme is simple, it cannot unambiguously tag consecutive mentions. Therefore, the `BIO` scheme [161], adopted by the CoNLL-2002 and CoNLL-2003 shared tasks [194, 195], is most often preferred:

Example from Edmond Hamilton's *The Star Kings*

B-PER	I-PER	O	O	O	B-PER	I-PER	O	B-PER	O	O	O	B-LOC	O	O	0
Shorr	Kan	said	to	bring	Zarth	Arn	and	Lianna	back	to	the	Cloud	at	once	.

In that scheme, tokens that *begin* an entity are tagged `B-TYPE`, which allows to unambiguously represent mention boundaries. Other formalizations of the NER problem exist. Notably, some researchers are interested in *nested NER*, a more general and harder version where mentions to detect can exist inside other mentions. In the following example, we can observe three layers of nesting[6]:

Example from the Litbank NER dataset [30], annotated from Jane Austen's *Emma*

B-PER	I-PER	I-PER	I-PER	I-PER	I-PER	I-PER	0
the	elder	brother	of	Isabella	's	husband	.
O	O	O	O	B-PER	I-PER	I-PER	0
the	elder	brother	of	Isabella	's	husband	.
O	O	O	O	B-PER	O	O	0
the	elder	brother	of	Isabella	's	husband	.

By opposition to nested NER, "traditional" sequence-labeling NER is sometimes termed *flat NER*, which will be what we refer to in this thesis when we generally mention NER.

NER SYSTEMS The first NER systems used rules and gazetteers [162]. Systems later progressed by using machine learning techniques [143] and then deep learning. Since the advent of BERT [60], NER has been marked by the usage of transformer-based models. More recently, LUKE [229], a transformer-based model specially constructed for entity-related tasks, has pushed performance further. As NER is a widely studied task, various other techniques are used depending on corpora and applications, and it would be impossible to list them all here. See Keraghel et al. [105] for a recent survey.

MEASURES The standard measure of performance for NER is F1-Score, which is the harmonic mean of Precision and Recall. These measures are only computed on gold and predicted mentions, and not on token tags themselves, as the `O` (outside mentions) token class constitutes the majority of tags in texts. The computation of these measures may seem trivial, but in practice great care must be taken when performing evaluation, as it is easy to obtain non-reproducible or misleading results [117]. A typical issue is the handling of invalid tagging output: for example, a model may incorrectly tag the start of an entity with an `I-TYPE` tag under the BIO scheme. Depending on the implementation, this may or may not be penalized when computing measures. To avoid such issues, it is better to use existing tools such as the official CoNLL scorer `conlleval.pl` or the `seqeval` [144] Python package, to ensure

6 In that example, we represent nesting with several tagging levels, but this can be cumbersome depending on the nesting level, so there exist other ways to represent nested NER. Yan et al. [230] propose a scheme to represent both flat and nested NER at the same time.

implementation choices are known and results comparable. We use `seqeval` ourselves in all the NER-related experiments described in this manuscript.

DATASETS Since NER has been studied extensively, there are many existing datasets, one of the most famous being CoNLL-2003 [195], consisting of short news stories. Despite its popularity, CoNLL-2003 has shortcomings: its documents are short, and there are some annotation errors [171, 210]. When it comes to the literary domain, there are only a few datasets, even in English. The Litbank dataset [30] annotates excerpts from 100 novels, but is concerned with nested NER. A French version of this dataset, fr-Litbank[7], also exists but contains fewer excerpts (18 vs. 100). To our knowledge, there only exists a few flat NER datasets in the literary domain: the English OWTO dataset of Dekker et al. [59], and the French dataset from Alrahabi et al. [6]. OWTO is composed of approximately the first chapter of 40 English novels, 20 of which are classical ones while the others are more recent. During this thesis, we surveyed this dataset [13, 17] and concluded that it had shortcomings: as a result, we propose the *Novelties* English dataset we present in Section 3.2.

3.1.3 *Coreference Resolution*

Coreference resolution is the task of linking together mentions of an entity in a text, which creates a *coreference chain* (sometimes also termed *mention cluster*) of mentions [157]. An example is shown in Figure 5. While NER detects alias mentions, coreference resolution can link these to generic mentions (pronouns, generic expressions...), which supports detecting more character mentions. This makes it an important step when extracting networks, as some interactions between characters can be missed when some character mentions are missing. Coreference can also be used as input when performing the character unification task we present in Section 3.1.4.

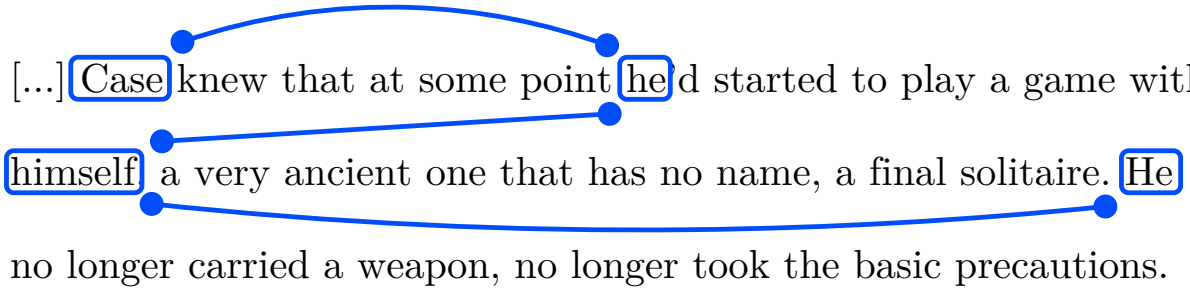


Figure 5.: Coreference resolution on an excerpt from William Gibson's *Neuromancer*. The blue coreference chain is composed of mentions of the main character, Case. NER would only detect the name "Case", but not the pronominal mentions.

Coreference resolution systems link mentions together, but linking mentions necessitates detecting them first. Some models assume that mentions are already detected beforehand and only predict links, but others (such as the end-to-end models based on Lee et al. [113, 114]) also perform *mention identification* prior to linking, which renders the task more difficult.

COREFERENCE SYSTEMS As surveyed by Poesio et al. [157], early coreference systems used linguistic knowledge to link mentions together. Gradually, systems shifted towards corpus-based statistical approaches with the advent of the MUC-6 and MUC-7 conferences. More recently, the seminal work of Lee et al. [113, 114] is an important foundation for modern

7 https://github.com/lattice-8094/fr-litbank

coreference resolution systems. The authors model coreference resolution as a mention ranking problem. They output a representation for each text span using a Bi-LSTM [92] and then compute a mention score for each span, denoting the probability of this span to be a mention. Then, they compute a pairwise score for each pair of spans. Based on the sum of mention scores and pairwise scores, an antecedent (or no antecedent if this sum is negative) is computed for each span, effectively creating coreference chains. Lee et al. [113, 114] model is said to be end-to-end, as the same model performs mention detection and mentions linking. Joshi et al. [102] later replace the Bi-LSTM with a SpanBERT encoder [101], improving performance. While these models perform well on benchmarks compared to previous ones, they have an important shortcoming. Given n the number of input tokens, the complexity of the model is $O(n^4)$, leading to computational and memory issues, especially when working on long documents. In practice, tricks are used to consider fewer pairs of spans and reduce complexity, such as pruning mentions based on their mention score to reduce the number of pairwise score computations. Nonetheless, researchers later tried to find ways to formalize the problem differently to avoid computational problems. Dobrovolskii [63] first works on pairs of tokens, later reconstructing spans from their head tokens. Another line of work explores incremental models, resolving chains left-to-right using a caching mechanism [86, 196, 226]. While their performance is not as good when compared to end-to-end models, their lower complexity makes them more easily applicable to long documents. Finally, other authors try to leverage generative models to cast the problem as sequence-to-sequence [40, 119, 201, 239].

MEASURES Evaluating coreference resolution is difficult in general. There are many existing measures (mainly MUC [204], B^3 [26], CEAF [124], BLANC [163] and LEA [137]), and none of these are entirely satisfying or measure the same thing. Even worse, many authors use *CoNLL-F1*, an average of three measures (MUC, B^3 and $CEAF_{\phi 4}$), to compare different systems. The three original measures already have shortcomings and, as Moosavi and Strube [137] rightfully state, *"averaging three unreliable scores does not result in a reliable one"*.

DATASETS There are multiple coreference resolution datasets to train and evaluate models. The most used resource is likely CoNLL-2012 [159][8], which is multilingual (English, Chinese and Arabic) and encompasses multiple domains (newswire, web, phone conversations...). Other well-known English ones include PreCo [49] and WikiCoref [80]. In the literature domain, however, only a few datasets exist. For English, Litbank has a coreference layer [29]. For French, the multi-domain dataset Democrat [111] contains a few literary texts, while fr-Litbank [112] adopts the Litbank annotation guidelines. All of these datasets only include short texts when compared to the scale of entire novels, which is an issue as coreference resolution on long documents remains a very challenging task [86, 87]. We are only aware of a few entire texts being annotated. In English, Guo et al. [86] annotate coreference for the top 20 characters of Orwell's *The Animal Farm*. In German, Krug [107] annotates Ludwig Tieck's short fairy tale *Der blonde Eckbert* and Theodore Fontane's novel *Effi Bries*.

3.1.4 *Character Unification*

Character unification can be considered as a global version of coreference resolution specific to characters. There are two goals for this task: first, determine characters and all of their aliases. Second, attribute each character mention in the text to a single character. We distinguish this task from coreference resolution because, in practice in Renard [20], we use the

8 CoNLL-2012 is a subset of the OntoNotes v5 dataset [213].

output of a coreference model as input to a second module to perform character unification. This approach is necessary as the performance of coreference resolution systems in long texts is low [86, 87], and is inspired by existing literature on the subject [202]. It allows linking mentions that are extremely distant and grammatically disconnected, which coreference systems struggle to do. The character unification task is not well-defined in the literature, and is sometimes termed differently. Alternative names include "character detection", "character clustering" or "alias resolution".

CHARACTER UNIFICATION SYSTEMS Most works performing character unification do not solve the task for itself, but rather as part of some other higher-level task. For example, Bamman et al. [31] perform "character clustering" when they need to group character aliases to classify characters, and Cuesta-Lazaro et al. [56] solve character unification to implement speaker attribution in novels. Other works are specifically concerned with character unification. Vala et al. [202] perform alias resolution using a set of rules, including the inspection of coreference resolution chains. They create a graph where vertices represent aliases, and add or remove edges between them according to these rules. After having applied all these, connected components form character alias clusters. Jahan et al. [99] propose a 2-stage system supposing the extraction of coreference chains. First, they determine which coreference chains concern animated entities using an off-the-shelf tool. Then, they train an SVM classifier to recognize whether chains concern characters. The quality of the initial coreference chains has a large impact on the final performance.

DATASETS There are no standard datasets or measures for character unification. However, datasets with other forms of annotations are sometimes usable to evaluate models. This is the case for Litbank [29], where coreference resolution and NER data can be combined to obtain character mentions. The PDNC speaker attribution dataset [205] can also be used, but only speakers are annotated. In terms of measures, Vala et al. [202] propose Character Precision and Character Recall, which we use when evaluating Renard in Section 3.4.1.

3.1.5 *Speaker Attribution*

The Commissioner put out a lower lip and suddenly looked like a child about to pout. **"But what's the point? What kind of information are you after?"**

"I'm not sure, Commissioner. I just have a feeling that in a case like this, information on robots might help." Baley clamped his mouth shut after that. He wasn't going to be specific, and that was that.

"I wouldn't, Lije. I wouldn't. I don't think it's wise."

"What's your objection, Commissioner?"

Figure 6.: Example of speaker attribution in Asimov's *The Caves of Steel*. Brown utterances are spoken by Commissioner Julius Enderby, while blue ones are spoken by the main character Elijah "Lije" Baley. For the first two, the speaker is said to be *explicit* as it is clearly mentioned in the text. For the last two quotes, speakers are *implicit*, which makes them comparatively harder to infer.

Speaker attribution (sometimes termed "quote attribution") is a fundamental task when it comes to extracting conversational networks. It consists in extracting the correct speaker for speech in a text, as seen in Figure 6. Speaker attribution first necessitates knowing where

speech appears in the text, which is the subtask known as *quote detection*. This subtask can be carried out using regular expressions or machine learning approaches [65].

The first speaker attribution systems used sets of rules to determine a speaker [82]. For example, the subject of speech verbs is an important cue for the speaker's identity. Subsequent works use machine learning, using features sometimes adapted from the rules previously employed [70, 90, 149, 234]. Later, Muzny et al. [141] propose a hybrid sieves-based system with rules and a machine learning model. Amalvy [7] use a BERT quote encoder [60] and a classification head to solve the task. Cuesta-Lazaro et al. [56] proceed similarly but add dialogue state tracking. Recently, Su et al. [189] propose to use LLMs to perform the task as sequence-to-sequence.

In terms of datasets and measures, Litbank [182] has a speaker attribution layer. Sims and Bamman [182] measure speaker attribution performance by adapting coreference resolution measures (MUC [204], B^3 [26] and $CEAF_{\phi_4}$ [124]). In 2022, Vishnubhotla et al. [205] released the PDNC corpus, composed of 22 novels with annotated quotations. PDNC measures speaker attribution using accuracy.

3.2 THE NOVELTIES DATASET

The OWTO[9] dataset of Dekker et al. [59] is, to our knowledge, the only English literary *flat* NER dataset. OWTO consists of approximately the first chapter of 40 novels, 20 of which are classic works while 20 are more recent. After experimentation, we conclude that this dataset has errors and shortcomings [13, 17]: for example, only `PER` entities are annotated. Therefore, we correct and extend it to propose the *Novelties* dataset.

3.2.1 *Novelties v0.1.0*

We find and fix issues of the original OWTO dataset. We give the full details of this correction process in Appendix A.1.1. We fix the following issues:

ANNOTATION ERRORS We find false negatives, false positives and inconsistencies. Inconsistencies consist of similar spans of text annotated differently for no apparent reason (for example, honorifics are annotated for some entities but not for others). We use a semi-automated process to correct these issues. First, we design a set of rules to prompt the annotator to verify some potential error cases. Then, we use a BERT-based model [60] trained on a slightly modified version of the CoNLL-2003 dataset [195] to highlight other potential errors. Finally, as a last resort, we manually correct the remaining errors.

ENCODING ISSUES Some characters are incorrectly represented (e.g. `d'Artagnan` was represented as `dâ€™Artagnan`). We manually fix these issues.

TOKENIZATION ISSUES We find inconsistencies in the dataset regarding tokenization, notably regarding apostrophed names: some are split while others are not. We manually correct this issue to enforce consistency.

QUOTING ISSUES Quoting is handled inconsistently. Depending on the novel, double quotes or simple quotes indicate dialogue. Backquotes are used to indicate quotation starts, but are sometimes wrongly placed at their end. Finally, some quotes are not found in pairs,

9 Short for *"Out With The Old and in with the novel"*

rendering dialogue identification difficult. We fix these issues semi-manually, by using an automated algorithm, and then fixing the remaining issues by hand.

The original dataset only consists of annotated PER entity mentions. We go further and also annotate locations (LOC) and organizations (ORG) by designing a short annotation guide (see Appendix A.1.1 for more details).

3.2.2 *Novelties v1.0.0*

While Novelties v0.1.0 represents an improvement over the original OWTO dataset [59], it has some limitations:

- Its guidelines are short and do not encompass all possible cases.
- Some mentions labeled as PER are not directly character mentions but, rather, mentions of a group of characters. This is the case, for example, for plural family names that we annotate as PER (like *The Proudfoots*).
- A single annotator annotated the entirety of the dataset.
- Novelties v0.1.0 is only composed of approximately a single chapter of each novel, and does not contain full books.

Due to these shortcomings, we propose Novelties v1.0.0. For this new and improved version of the dataset, we create detailed NER annotation guidelines [10]. We add two new types of mentions: *groups* (GRP) and *miscellaneous* (MSC). GRP represent groups of characters, which allow us to more precisely extract characters themselves. Meanwhile, miscellaneous mentions gather categories such as cultural or intellectual works, praxonyms (names referring to historical, cultural, commercial or sport events), unique ergonyms (names of objects and man-made products) and phenonyms (names of meteorological events such as tempests). To mark annotation differences with guidelines mentioning the PER class, we change to name of the class to CHR (CHaRacter). Finally, we annotate entire novels, as opposed to single chapters, and employ two annotators to compute inter-annotator agreement on a subset of the dataset. To ensure high quality, after computation of the agreement as an indication, we fix disagreements and incorporate the fixed annotations in the dataset.

Additionally, we create guidelines for the character unification (also known as alias resolution) task [11], and annotate our full novels with them.

Using this new annotation process, we produce a new dataset of 11 entire novels annotated for NER and character unification, and NER annotations for chapters from 8 other novels. See Table 1 for details. As the original OWTO [59], Novelties sports a mix of classical novels as well as more recent ones, including fantasy and science-fiction. For NER, we compute agreement on 5 chapters shared between the two annotators of the corpus, and report pairwise F1-score and two variations of Cohen's Kappa [52]: one computed on all tokens, and one computed on tokens annotated as an entity mention by at least one annotator. We do so because computing Cohen's Kappa on all tokens outputs an inflated score, as the majority class of tokens is very largely O (i.e. tokens are mostly not parts of entity mentions). We find a high agreement between annotators, with a pairwise F1-score of 92.58, Cohen's Kappa computed on all tokens of 94.20 and Cohen's Kappa computed on annotated mentions of 85.17.

Novel	No. NER annotated chapters	character unification
1984	24 (full)	yes
A Game of Thrones	1	no
Alice in Wonderland	3	no
Assassin's Apprentice	1	no
A Study in Scarlet	1	no
Bel Ami	18 (full)	yes
Black Prism	1	no
Brave New World	18 (full)	yes
David Copperfield	1	no
Dracula	1	no
Eugenie Grandet	14 (full)	yes
Germinal	40 (full)	yes
Madame Bovary	35 (full)	yes
Moby Dick	136 (full)	yes
The Black Company	7 (full)	yes
The Blade Itself	46 (full)	yes
The Red and The Black	75 (full)	yes
The Three Musketeers	68 (full)	yes

Table 1.: Annotated novels in Novelties v1.0.0

3.3 RENARD: A CHARACTER NETWORK EXTRACTION PIPELINE

3.3.1 *Motivation*

There are several existing extraction pipelines when it comes to character networks. We list two types of pipelines: general NLP pipelines and character network extraction pipelines.

General NLP pipelines are "general" in the sense that they are not specific to character network extraction, but they are used for many other tasks. General pipelines perform NLP tasks to annotate a text with information (NER, coreference resolution), and researchers use these annotations to extract a network. Among the most used NLP pipelines such as NLTK [38] or spaCy [95], two are common choices when extracting character networks: Stanford CoreNLP [131] and BookNLP [31]. BookNLP in particular is specifically designed for novels, making it a popular choice.

Character network extraction pipelines are less commonly used, but can directly extract character networks. Among these, SINNET [3] can extract directed networks of "social events" comprised of interactions and observations. CHAPLIN [186] is a semi-automatic pipeline that can extract weighted networks from theater plays and novels. rCat [34] is a pipeline with a web interface targeted towards humanities researchers. Tex2net allows extracting networks where each edge is a list of events along with their temporality, aimed at efficiently summarizing stories. Finally, Charnetto [142] can extract networks from books and movie scripts using either Flair [5] or SpaCy [95].

While some of the existing general NLP pipelines are modular, they are not designed to extract character networks, which means additional effort is required to do so. Except BookNLP, they are not directly concerned about literary texts. Meanwhile, existing character network extraction pipelines do not support extracting dynamic networks, are not modular and usually target a single type of networks (co-occurrences, events...).

All of these drawbacks lead us to propose Renard [20], a modular extraction pipeline that can extract static and dynamic networks from narrative texts. We release Renard under a free license, and make it available as a Python package[10] (`pip install renard-pipeline`). We base Renard on the generic character network extraction framework we present in Section 2.2. We design it so that it is as modular as possible, which allows the user to select the implementation of each extraction step as needed. This has several advantages:

1. Depending on the input text, the user can choose the most relevant series of steps and configure each of them as needed. Therefore, the pipeline can be specialized for different types of texts, allowing for better performance.

2. The pipeline can easily incorporate new advances in NLP, by simply implementing a new step when necessary.

3. Crucially, one can study the impact of the performance of each step on the quality of the extracted networks, and on the performance of downstream tasks.

We intend for Renard to be used by digital humanities as well as NLP researchers and practitioners. The former category of users can use Renard to quickly extract character networks for literary analysis. Meanwhile, the latter can use Renard to easily represent textual content using networks, which can be used as inputs for downstream tasks (classification, recommendation...). They can also use Renard to study the impact of NLP task errors on the extracted networks.

10 https://github.com/CompNet/Renard

3.3.2 *Design and Main Features*

Renard is centered on the concept of a *pipeline*. In Renard, a pipeline is a series of sequential *steps* that are run one after the other in order to extract a character network from a text. Steps generally correspond to the tasks we presented in Section 3.1, except for preprocessing, tokenization and network extraction steps. Preprocessing steps simply prepare the text better for subsequent steps. Tokenization cuts the text into tokens and sentences, and is one of the first steps to run in any text processing pipeline. Network extraction steps are always last in a pipeline, and use information from the previous steps to construct a character network. These last network extraction steps are purely deterministic: for example, extracting a co-occurrence network given characters and their occurrences is a matter of a simple algorithm.

In practice, when using Renard, the user simply *describes* a pipeline in Python by specifying a series of steps, and can apply it to different texts afterward. The following code block exemplifies that philosophy:

```
from renard.pipeline import Pipeline
from renard.pipeline.tokenization import NLTKTokenizer
from renard.pipeline.ner import NLTKNamedEntityRecognizer
from renard.pipeline.character_unification import GraphRulesCharacterUnifier
from renard.pipeline.graph_extraction import CoOccurrencesGraphExtractor

with open("./my_doc.txt") as f:
    text = f.read()

pipeline = Pipeline(
    [
        NLTKTokenizer(),
        NLTKNamedEntityRecognizer(),
        GraphRulesCharacterUnifier(min_appearances=10),
        # users can pass dynamic=True and specify the
        # dynamic_window argument to extract a dynamic network
        # instead of a static one.
        CoOccurrencesGraphExtractor(
            co_occurrences_dist=(10, "tokens"), dynamic=False
        )
    ]
)

out = pipeline(text)
```

As an example, Figure 7 shows the co-occurrence character network of Jane Austen's "Pride and Prejudice", extracted using the Renard pipeline above. While this network is static, users can also extract a dynamic network by passing the `dynamic=True` argument to the last step of the pipeline, and specifying the `dynamic_window` argument: in that case, Renard outputs a list of graphs corresponding to a dynamic network instead of a single network[11]. Renard uses the NetworkX Python library [89] to manipulate graphs, ensuring compatibility with a wide array of tools and formats.

To allow for custom needs, we design Renard to be very flexible. If a step is not available in Renard, we encourage users to either:

11 See the documentation on dynamic networks for more details: https://compnet.github.io/Renard/pipeline.html#dynamic-graphs

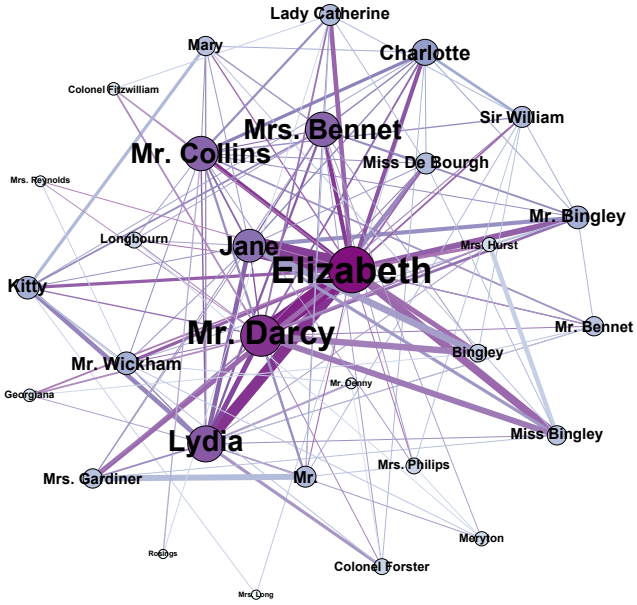


Figure 7.: Co-occurrence character network of Jane Austen's "Pride and Prejudice", extracted automatically using Renard. Vertex size and color denote degree, while edge thickness and color denote the number of co-occurrences between two characters.

- Externally perform the computation corresponding to the desired step, and inject the results back into the pipeline at runtime,
- Implement their own step to integrate their custom processing into Renard by subclassing the existing `PipelineStep` class. If necessary, this `PipelineStep` can act as an adapter to an external process that may or may not be written in Python.

The flexibility of this approach introduces the possibility of creating invalid pipelines because steps often require information computed by previously run steps: for example, solving the NER task requires a tokenized version of the input text. To counteract this issue, each step therefore declares its requirements and the new information it produces, which allows Renard to check whether a pipeline is valid, and to explain at runtime to the user why it may not be[12]. When a user implements a custom step, he can specify this information to allow Renard to perform validation.

Renard lets users select the targeted language of their custom pipelines. A pipeline can be configured to run in any language, as long as each of its steps supports it. Table 2 shows all the currently available steps in Renard and their supported languages.

3.3.3 *NLP Tasks Implementation*

In this section, we quickly present the implementation of the main NLP tasks supported by Renard. We restrain this presentation to the more challenging steps we implemented.

NER: `BertNamedEntityRecognizer`

We implement a typical NER model, where we use a BERT [60] encoder to compute the representation of each token, and a fully connected classification head to perform classification

12 See the documentation for more details on steps requirements: https://compnet.github.io/Renard/pipeline.html#the-pipeline

Step	Task	Supported Languages
`StanfordCoreNLPPipeline`	multiple	eng
`CustomSubstitutionPreprocessor`	preprocessing	any
`NLTKTokenizer`	tokenization	eng, fra, rus, ita... (13 more)
`QuoteDetector`	quote detection	any
`NLTKNamedEntityRecognizer`	NER	eng, rus
`BertNamedEntityRecognizer`	NER	eng, fra
`BertCoreferenceResolver`	coreference resolution	eng
`SpacyCorefereeCoreferenceResolver`	coreference resolution	eng
`NaiveCharacterUnifier`	character unification	any
`GraphRulesCharacterUnifier`	character unification	eng, fra
`BertSpeakerDetector`	speaker attribution	eng
`CoOccurencesGraphExtractor`	network extraction	any
`ConversationalGraphExtractor`	network extraction	any

Table 2.: Existing steps and their supported languages in Renard. Languages are named according to their ISO-639-3 code.

on each token. We finetune that NER model on the Novelties v0.1.0 dataset we introduced in Section 3.1.2.

Coreference Resolution: `BertCoreferenceResolver`

We base our coreference module on the end-to-end system of Lee et al. [113]. We reimplement their system using PyTorch, and add a few enhancements:

- We use a pretrained BERT encoder [60] instead of a Bi-LSTM, as Joshi et al. [102] show that it yields to better performance.
- We support singleton mentions by using an additional mention loss, as described by Xu and Choi [228].
- We support hierarchical merging on long texts, as introduced by Gupta et al. [87]. Hierarchical merging allows processing long documents with less computational power and memory by computing coreference resolution on blocks of texts, and then recursively merging coreference chains from adjacent blocks.

Considering the limited number of freely available easy-to-use coreference models, we provide our coreference system as an independent open release under the name *Tibert* [18][13]. Tibert allows to easily perform coreference resolution inference, but also to train a model with a few lines of codes, and already supports three datasets: Litbank [29], fr-Litbank[14] and WikiCoref [80]. We make Tibert available as a Python package on PyPI (`pip install tibert`).

13 https://github.com/CompNet/Tibert
14 https://github.com/lattice-8094/fr-litbank

Speaker Attribution: `BertSpeakerDetector`

We implement a speaker attribution module based on the speaker rating model we proposed previously [7]. Since freely available speaker attribution models are rare, we freely release this system separately under the name *Grimbert* [15]. We also freely release a model trained on PDNC [15].

Our model takes as input a quote q_t and a set P of candidate speakers, and predicts speaker y_t. To do so, the model computes a score $s_{t,i}$ between 0 and 1 for each candidate speaker p_i: that score represents the probability $P(y_t = p_i)$ of p_i being the speaker of quote q_t. We can then predict $y_t = argmax_i(s_{t,i})$. In cases where $max_i(s_{t,i}) < 0.5$, we consider that the model is unsure of its prediction, and we predict "unknown speaker".

In order to compute score $s_{t,i}$, we first produce a representation of quote q_t and of each candidate speaker p_i. To do so, we embed the context surrounding the quote (including the quote itself) using a SpanBERT encoder [101]:

$$SpanBERT(o_{t,l}...o_{t,n}) = \mathbf{e_{t,l}}...\mathbf{e_{t,n}} \tag{6}$$

We take the delimiter of quote q_t in its context to produce its representation $f(q_t) = [\mathbf{e_{t,k}}; \mathbf{e_{t,l}}]$, where ; denotes concatenation. We compute the representation of a candidate speaker p_i as the average embedding of their mentions in context: $g(q_t, p_i) = mean_j(\mathbf{m_{i,j}})$.

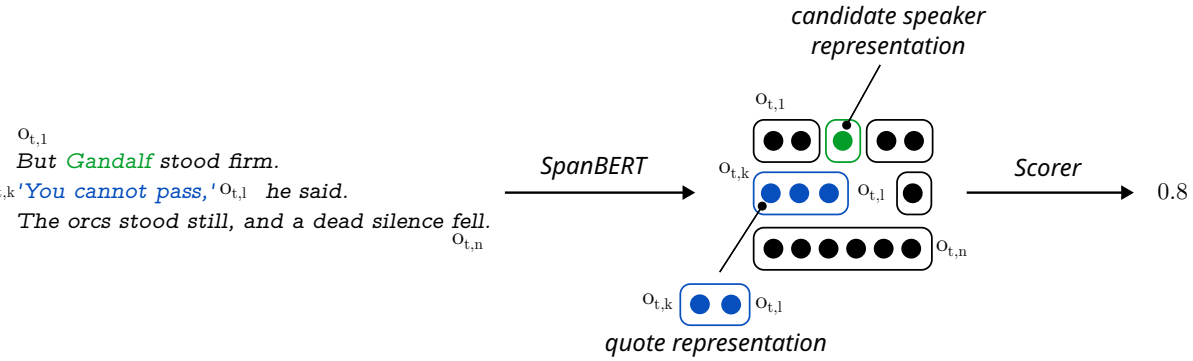


Figure 8.: Scoring process for quote *"You cannot pass"* and candidate speaker *"Gandalf"*.

Using these two representations, we compute $s_{t,i}$ as the score of the positive probability class calculated using a feed-forward network (FFN) followed by a softmax:

$$s_{t,i} = softmax\Big(FFN\big([f(q_t); g(q_t, p_i)]\big)\Big)_{pos} \tag{7}$$

Figure 8 shows the process of scoring a candidate speaker for a specific quote.

Character Unification: `GraphRulesCharacterUnifier`

We base Renard's character unification step on the rule-based system inspired by Vala et al. [202]. We keep some rules as-is, but adapt and delete some others. The module works by first creating a graph, where each vertex is a unique alias extracted by a previous NER step. Then, we use a set of rules to connect or disconnect two different aliases in this graph. After having applied all rules, we merge the connected components to obtain sets of aliases $\{S_1, ..., S_n\}$ that hopefully correspond to characters. We use the following rules:

1. When two aliases have a first or last name in common, connect them (*"Emma"* and *"Emma Woodhouse"*).

15 https://huggingface.co/compnet-renard/spanbert-base-cased-literary-speaker-attribution

2. When two aliases are related according to a hypocorism gazetteer (*"John"* and *"Johnny"*), connect them.

3. When one of the two above rules holds when removing honorifics, connect them (*"Mr. John"* and *"Johnny"*).

4. When connected aliases have the same last name but a different first name, delete all vertices in the shortest paths between them (*"John Traitor"* and *"Smith Traitor"*).

5. When two names have a different inferred gender, delete all the edges in the shortest paths between them (*"Mr. Traitor"* and *"Miss Traitor"*). We infer gender using honorifics, and coreference chains when they are available.

6. When coreference information is available, we connect two aliases if they are deemed to be coreferential. We consider two aliases to be coreferential when they appear together in one or more coreference chains, and never appear without the other in other chains.

3.4 EVALUATION AND ERROR ANALYSIS

Since the quality of the networks extracted by our pipeline is important for applications, in this section, we strive to understand the performance of Renard. We also shed light on the impact of NER and coreference resolution errors on the extracted networks, in order to guide future research on improving extraction.

3.4.1 *Evaluation Measures*

To understand the quality of networks extracted by Renard, we need a set of evaluation measures. We base our measures on the work of Vala et al. [202] on alias resolution.

Let $G_p = (V_p, E_p)$ be a predicted character network, and $G_g = (V_g, E_g)$ be the corresponding gold network. Let each vertex of V_p and V_g represents the set of aliases $\{a_1, a_2, \cdots, a_n\}$ of the underlying character. In order to know whether the predicted network G_p correctly contains vertices and edges similar to the gold network G_g, we first need to match their characters, since a vertex in V_p is not necessarily present in V_g and vice versa. Thus, we start by computing a mapping f_V from the set of predicted vertices V_p to $V_g \cup \{v_\varnothing\}$. This mapping associates any predicted vertex $u \in V_p$ to a gold vertex $v \in V_g$ or to the null vertex $v_\varnothing$, meaning u is not associated with any character in V_g. Note that the null vertex $v_\varnothing$ represents the empty ensemble of aliases. Symmetrically, we construct a mapping g_V from V_g to $V_p \cup \{v_\varnothing\}$. As Vala et al. [202], we compute these mappings by means of maximum bipartite matching. We leverage the alias sets represented by the vertices to compute Vertex Precision Pre_V and Vertex Recall Pre_R[16]:

$$Pre_V = \max_{f_V} \frac{\sum_{u \in V_p} 1 - \frac{|u - f_V(u)|}{|u|}}{|V_p|} \tag{8}$$

$$Rec_V = \max_{g_V} \frac{\sum_{v \in V_g} [g_V(v) \cap v \neq v_\varnothing]}{|V_g|}. \tag{9}$$

Measure Vertex F1 ($F1_V$) is the harmonic mean of Vertex Precision and Vertex Recall.

16 We employ the Iverson bracket notation, where $[P] = 1$ if proposition P is true, and 0 otherwise.

For edges, we use mappings f_V and g_V to construct mappings f_E and g_E:

$$f_E(\{u,v\}) = \begin{cases} \{f_V(u), f_V(v)\}, & \text{if } f_V(u) \neq v_\varnothing \text{ and } f_V(v) \neq v_\varnothing \\ e_\varnothing, & \text{otherwise,} \end{cases} \tag{10}$$

with $e_\varnothing$ being the null edge. Finally, we compute Edge Precision and Edge Recall:

$$Pre_E = \max_{f_E} \Big(\frac{\big|\{f_E(e) : e \in E_p\} \cap E_g\big|}{|E_p|} \Big) \tag{11}$$

$$Rec_E = \max_{g_E} \Big(\frac{\big|E_p \cap \{g_E(e) : e \in E_g\}\big|}{|E_g|} \Big). \tag{12}$$

We define Edge F1 ($F1_E$) as the harmonic mean of Edge Precision and Edge Recall.

We also introduce weighted variants of these network measures ($WPre_E$, $WRec_E$ and $WF1_E$) in order to take into account the weights of the network edges. Each weight corresponds to the number of interactions between the connected characters. Before computing the measures, we normalize the weights by dividing by the maximum edge weight in the network. We compute Precision and Recall as follows:

$$WPre_E = \max_{f_E} \Big(\frac{\sum_{e \in E_p} 1 - \big|w(f_E(e)) - w(e)\big|}{|E_p|} \Big) \tag{13}$$

$$WRec_E = \max_{g_E} \Big(\frac{\sum_{e \in E_g} 1 - \big|w(e) - w(g_E(e))\big|}{|E_g|} \Big), \tag{14}$$

where $w(e)$ is the function that computes the normalized weight of edge e. $w(e)$ is 0 when $e = e_\varnothing$. Weighted measures evaluate the quality of the distribution of weights in the predicted network, and are always less than or equal to their unweighted counterparts.

3.4.2 *Evaluation Corpus*

In the remainder of this chapter, we perform all of our experiments on the Litbank literary corpus [29, 30]. We use the NER and coreference resolution layers of the dataset. Since Litbank is designed for nested NER while the Renard NER step performs flat NER, we flatten the Litbank annotations using an automated algorithm. We use coreference chains of `PER` entities as the ground truth for the character unification step, since coreference resolution on characters is equivalent to character unification. We extract gold character networks using these annotations and the graph extraction step.

Litbank is composed of excerpts from 100 novels of approximately 2,000 tokens each. Since these excerpts can be short, we restrict our analysis to the 30 novels involving at least 10 characters. This prevents high deviation of network quality measures when a vertex or an edge is modified in the prediction. We use the remaining 70 novel excerpts to train coreference resolution models: we use 63 of these 70 novels (90%) as a training set, and the remaining 7 as a development set.

3.4.3 *Pipeline Performance*

Experiments

We apply Renard to the 30 excerpts we select for analysis and extract co-occurrence networks. Since multiple coreference resolution measures exist and none of these are entirely satisfying

or measure the same thing, we report a large set of measures including MUC [204], B^3 [26], CEAF [124], BLANC [163] and LEA [137]. We also report the performance of a pipeline without the optional coreference resolution step, in order to assess the usefulness of the task. We consider a co-occurrence window of 32 tokens. To be precise, we evaluate the following Renard pipelines:

```
# w corefs
Pipeline(
    [
        NLTKTokenizer(),
        BertNamedEntityRecognizer(),
        BertCoreferenceResolver(),
        GraphRulesCharacterUnifier(link_coref_mentions=True),
        CoOccurrencesGraphExtractor(co_occurrences_dist=(32, "tokens"))
    ]
)

# w/o corefs
Pipeline(
    [
        NLTKTokenizer(),
        BertNamedEntityRecognizer(),
        GraphRulesCharacterUnifier(),
        CoOccurrencesGraphExtractor(co_occurrences_dist=(32, "tokens"))
    ]
)
```

Results

Table 3 shows the NER and coreference resolution performance of our Renard pipeline. NER performance is below the reported state-of-the-art on datasets from other domains such as CoNLL-2003 [195] where the best systems obtain F1 scores higher than 90. While part of this lack of performance may be due to the way we transform the nested Litbank dataset to a flat NER dataset, we observe a high disparity of performance between novels, with an F1 ranging from 51.16 to 97.30. This matches our observations [16] and the ones from Dekker et al. [59], indicating challenges that are specific to some novels. We see examples of these challenges in Chapters 4 and 5. Meanwhile, the performance of coreference resolution is lower than what Bamman et al. [30] report on a 10-fold experiment on the same corpus. This may be due to the lower number of excerpts in our training set (63 vs. 80), which is required for our analysis.

Table 4 shows the performance of the Renard pipeline on our test corpus of 30 excerpts, depending on whether we add the optional coreference step or not. Vertex and Edge F1 are higher when omitting coreference information, likely because the performance of the coreference resolution algorithm is not high enough, leading to the detection of spurious mentions, which misleads both the character unification and co-occurrence detection steps. In general, edge recall is quite low, meaning a lot of interactions between characters are missed.

Meanwhile, using coreference information proves to be important to increase weighted edge measures. Even though coreference resolution leads to a compromised network structure overall, it allows the pipeline to detect more character mentions, leading to a better estimation of the relative strength of their interactions.

Task	Measure	Mean	Min	Max
NER	F1	79.58	51.16	97.30
Coref	MUC	78.45	64.31	88.75
	B_3	54.87	41.53	70.40
	CEAF	47.04	34.31	59.75
	BLANC	60.82	40.64	79.82
	LEA	28.84	19.44	43.95

Table 3.: Renard pipeline performance on NER and coreference resolution. We compute the Mean, Min and Max values over the series formed by the measures of the 30 novel excerpts of our analysis set.

Measure	w/ coref	w/o coref
$F1_V$	57.64	**70.39**
$F1_E$	40.19	**44.93**
$WF1_E$	**33.53**	30.55
Pre_V	59.32	**68.99**
Pre_E	48.07	**62.07**
$WPre_E$	38.40	**39.91**
Rec_V	57.77	**74.00**
Rec_E	**39.37**	37.81
$WRec_E$	**33.17**	26.18

Table 4.: Renard performance on network extraction with or without the coreference resolution step.

3.4.4 *Comparison with Large Language Models*

Experiments

Since Renard is a pipeline, errors may cascade and degrade performance. Meanwhile, end-to-end approaches are less modular and explainable, but are not subject to cascading errors by design. Therefore, inspired by the recent advances in LLMs, we survey their capability to be used as character network extractors, and compare them to Renard.

LLM-COREF We remark that the last network extraction step, co-occurrence detection, is completely deterministic: we simply add an edge between two characters if one of their mentions is in the chosen co-occurrence window. Therefore, in the case of a co-occurrence network, we can define the extraction problem as span extraction, where each extracted span must be assigned to the correct character. This problem definition could also be viewed as a coreference resolution on characters only. To tackle the problem this way, we prompt LLMs to mark character mentions in the text with a unique character ID.

LLM-E2E Given an input text, we simply prompt LLMs to produce the corresponding character network in a simplified version of the XML-based Graphml format.

We use few-shot prompting by providing examples of the task to the model. Appendix A.1.2 shows the exact prompt we use for *LLM-Coref* and *LLM-E2E*. For both methods, we make an effort to parse the LLM output in order to fix the slightly incorrect output format. We survey two proprietary models, GPT3.5 Turbo [46][17] and GPT4o[18], and a recent open weights model, Llama3-8b-instruct [197]. We compare our LLM-based models with the Renard pipeline without coreference information, as described in Section 3.4.3.

Results

Results of our extraction methods *LLM-Coref* and *LLM-E2E* can be found in Table 5.

	LLM-Coref			*LLM-E2E*			*Renard*
Measure	**Llama3**	**GPT-3.5T**	**GPT4o**	**Llama3**	**GPT-3.5T**	**GPT4o**	
$F1_V$	37.93	28.99	52.32	56.87	44.26	62.98	**70.39**
$F1_E$	23.20	16.96	38.85	29.35	20.91	30.42	**44.93**
$WF1_E$	15.17	11.39	**32.72**	20.17	14.33	24.35	30.55
Pre_V	42.86	52.50	68.78	67.96	61.04	**77.96**	68.99
Pre_E	57.85	**65.78**	62.62	59.01	64.39	65.76	62.07
$WPre_E$	34.69	37.25	51.46	40.56	44.75	**53.97**	39.91
Rec_V	25.12	22.15	32.28	53.18	37.87	34.24	**70.39**
Rec_E	10.34	9.18	23.38	21.32	13.24	13.12	**37.81**
$WRec_E$	6.86	6.34	19.77	14.86	9.14	10.44	**26.18**

Table 5.: Comparison between the network extraction performance of LLM-based extraction methods and our Renard pipeline. *Llama3* stands for Llama3-8b-instruct, *GPT-3.5T* for GPT-3.5 Turbo.

LLM-COREF If we observe F1 scores, GPT4o performs the best amongst the LLMs we survey, followed by Llama3-8b-instruct and GPT-3.5 Turbo. LLMs particularly struggle with recall, with GPT3.5 Turbo and Llama3-8b-instruct missing a lot of character occurrences. Qualitatively, both of these models either miss a lot of mentions, or start hallucinating a lot of mentions before continuing that way through the output, influenced by their previous predictions. Llama3-8b-instruct also has trouble respecting the output format, sometimes generating invalid output. GPT4o is much more consistent, although it still misses a high number of mentions. When compared against the Renard pipeline, LLMs results are generally lower, except for Edge Precision and for weighted edge measures where GPT4o trades precision for recall.

LLM-E2E Results are generally higher than *LLM-Coref* method, highlighting the importance of the output format. If we focus on F1 scores, we observe the same ranking between LLMs, with GPT4o beating Llama3-8b-instruct and GPT3.5 Turbo. Generally, LLMs display high vertex and edge precision, even surpassing the pipeline sometimes. However, their recall still lags behind Renard.

17 GPT3.5 checkpoint: `gpt-3.5-turbo-0125`
18 GPT4o checkpoint: `gpt-4o-2024-05-13`

3.4.5 *NER Impact Analysis*

Experiments

To assess the impact of NER errors, we propose to start from a pipeline with gold NER predictions. As an example, we may start with the following predictions:

Gold Predictions Example

B-PER	O	O	O	O	O	O	O	B-PER	O
One-Eye	pranced	over	and	took	a	poke	at	Goblin	.

Then, we progressively degrade the performance of NER while observing the impact on the quality of the extracted networks. To degrade task performance, we add uniformly distributed perturbations to the predictions corresponding to different types of errors. We employ the two following NER perturbations:

ADD SPURIOUS NER MENTION We add a random false positive to the NER prediction by uniformly sampling a non-entity span (up to a certain span size) from the text, to reduce precision.

Add Spurious NER Mention: Example

B-PER	O	B-PER	O	O	O	O	O	B-PER	O
One-Eye	pranced	over	and	took	a	poke	at	Goblin	.

REMOVE CORRECT NER MENTION We remove a true positive from the NER predictions by uniformly sampling from the predicted NER mentions, to reduce recall.

Remove Correct NER Mention: Example

B-PER	O	O	O	O	O	O	O	O	O
One-Eye	pranced	over	and	took	a	poke	at	Goblin	.

Results

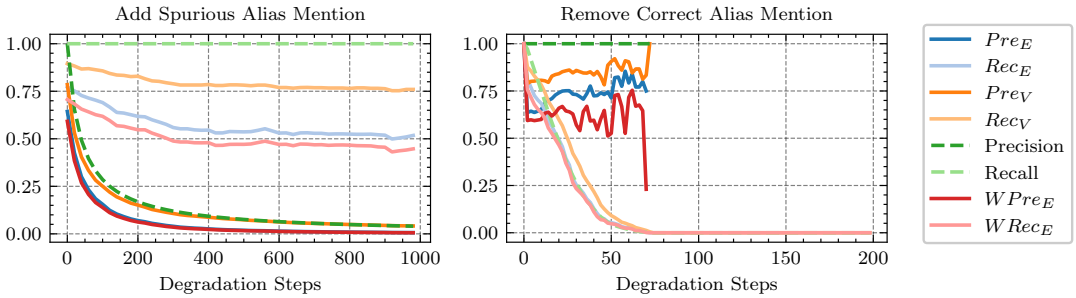


Figure 9.: Network measures versus number of degradation steps for "add spurious NER mention" and "remove correct NER mention" perturbations.

Results of applying NER perturbations can be found in Figure 9.

Unsurprisingly, reducing NER Precision has a direct effect on Vertex Precision, which plummets as more NER false positives are added. Edge Precision also sharply decreases as a result, while Vertex and Edge Recall slowly decrease down to a plateau. Meanwhile, reducing NER Recall sharply lowers network Recall measures, but Vertex Precision is not affected. Edge Precision measures become undefined after a while.

As we could intuitively expect, NER performance has a high impact on network quality. Since NER performance varies greatly depending on the novels as we note in Section 3.4.3, enhancing performance for challenging novels is an important concern that should be addressed by future research.

3.4.6 *Coreference Resolution Impact Analysis*

Experiments

To understand the impact of coreference resolution errors, we proceed as in NER. We start from gold predictions (colors indicate coreference chain):

Gold Predictions Example
1 One-Eye pranced over and took a poke at 2 Goblin, trying to break 2 his concentration.

And slowly apply perturbations while observing network quality measures. We employ the four following coreference resolution perturbations:

ADD SPURIOUS MENTIONS We add singletons (mentions linked to no other mentions) to the predictions consisting of incorrect mentions, by uniformly sampling non-mention spans (up to a certain span size).

Example
1 One-Eye pranced over and took a 3 poke at 2 Goblin, trying to break 2 his concentration.

REMOVE CORRECT MENTIONS We remove a correctly predicted mentions from the predictions by uniform sampling.

Example
One-Eye pranced over and took a poke at 2 Goblin, trying to break 2 his concentration.

ADD SPURIOUS LINKS We add incorrect coreference links between two mentions, wrongly merging coreference chains together. We uniformly sample the incorrect links in the set of all possible incorrect links.

Example
2 One-Eye pranced over and took a poke at 2 Goblin, trying to break 2 his concentration.

REMOVE CORRECT LINKS We remove correct links between predicted mentions, wrongly splitting coreference chains. We uniformly sample links among all existing correct coreference links.

Example
1 One-Eye pranced over and took a poke at 2 Goblin, trying to break 3 his concentration.

Results

The results of applying coreference resolution perturbations can be found in Figure 10.

ADD SPURIOUS MENTIONS Adding spurious singletons does not affect our character unification algorithm, and therefore has no impact on the quality of the extracted network.

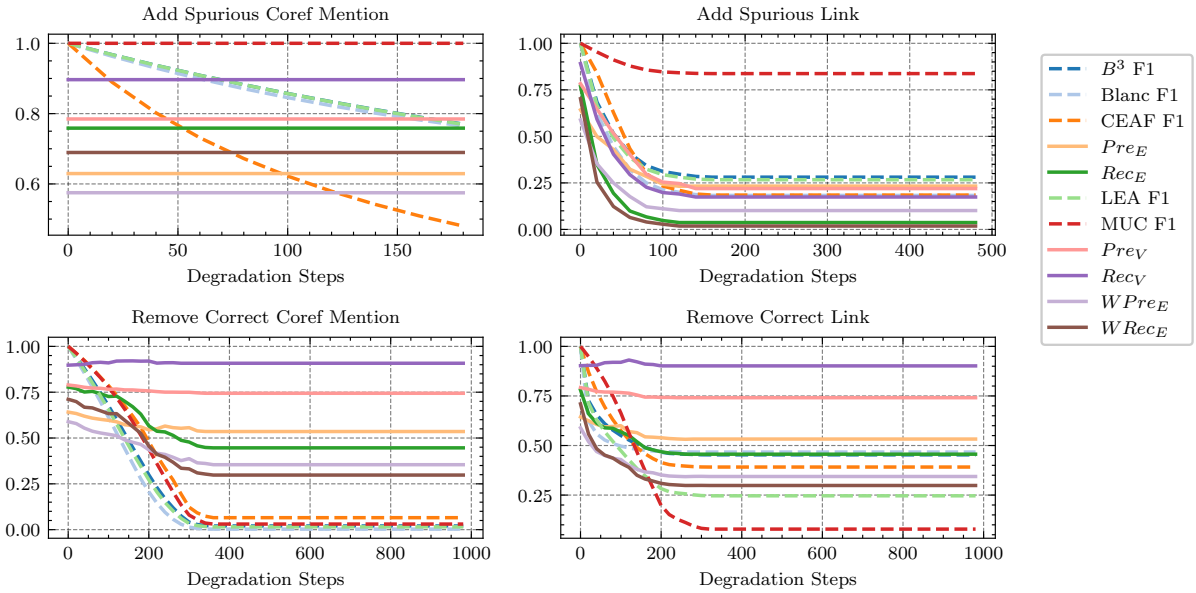


Figure 10.: Network measures versus the number of coreference resolution degradation steps.

REMOVE CORRECT MENTIONS Removing correct mentions mainly affects edge measures: characters are still recognized correctly, but some co-occurrence interactions are lost due to missing character mentions, leading to fewer edges.

ADD SPURIOUS LINKS Adding wrong coreference links sharply decreases all network extraction performance measures: characters are harder to recognize, and interactions are missing. This is driven by rules 5 and 6 of our character unification step (Section 3.3.3), that are both fed wrong information. While it would be possible to not apply these rules, rule 6 is the only one that allows linking two mentions with completely different forms.

REMOVE CORRECT LINKS Network edge measures are affected the most by coreference links removal, while vertex measures stay somewhat stable.

Not all coreference resolution errors prove to be equal in terms of impact on network quality. Adding spurious links is the most harmful degradation, while adding spurious singletons does not affect our character unification algorithm. Meanwhile, other errors mainly impact edge performance measures. Therefore, if one concern is to extract characters only, a conservative coreference algorithm with low linking recall but high linking precision might give sufficient performance. However, when extracting edges between characters, both high precision and recall are necessary, in terms of mentions as well as linking.

3.5 CONCLUSION

In this chapter, we presented the NLP tasks needed to extract character networks from textual content. We described the existing extraction systems. Noting some of their shortcomings, we implement Renard, a modular extraction pipeline that can be leveraged to study the impact of NLP task errors on the extracted networks. To evaluate Renard, we propose new network evaluation measures (Vertex F1, Edge F1 and Weighted Edge F1). Then, to better understand the weak points of pipeline-based approaches, we conducted a detailed study of the impact of NER and coreference errors on the extracted network. Through this study, we saw that NER performance is crucial to character network extraction. Since NER performance depends

strongly on the studied novel, we posit that understanding the specifics of novels and fixing the related errors is one of the next steps to improve character network extraction. We also observed that some errors in coreference resolution are more impactful than others. In order to predict the correct character aliases, linking precision is more important than linking recall, so developing a conservative linking algorithm is a possible next step to improve extraction. However, to extract edges, linking precision and recall are both necessary.

In the two following chapters, we try to improve NER performance in order to extract networks of higher quality. We first show how to increase recall on unconventional names in Chapter 4, and then work around the range issue of transformers-based models in Chapter 5.

4

DATA AUGMENTATION FOR DETECTING NAMES OF UNSEEN STYLES

Contents

4.1 Unseen Name Styles 44
4.2 Cross-Domain NER and Mention Replacement 44
4.3 Experiments 45
4.3.1 Mention Replacement 45
4.3.2 Datasets 47
4.3.3 Training and Evaluation 47
4.4 Results 47
4.4.1 Class Imbalance 49
4.4.2 Name Variety and Ambiguous Occurrences 49
4.5 Adaptation from Literary Texts to a Specific Genre 50
4.5.1 Results 51
4.6 Conclusion 51

In the previous chapter, we show that NER performance is crucial when it comes to extracting character networks. We also highlight that it varies a lot between novels, which indicates specific challenges for some of them. In this chapter, we tackle one of these challenges: unseen name styles. We start from the article of Dekker et al. [59], where they remark that mentions with some specific names from the literary domain are harder to detect for standard NER models. The models they survey are trained on data from a different domain (often on the popular newswire dataset CoNLL-2003 [195]), where some types of names do not appear. This issue is particularly marked for fantasy texts, where name styles can be specific and greatly differ from one lore to another. In the context of character network extraction, not detecting an alias mention because of its form is problematic, as it can lead to missing a character completely.

This name detection issue is a specific instance of the *cross-domain NER* problem: in this setup, a model is trained on a dataset from a *source domain* (in the case of Dekker et al. [59], newswire texts), and evaluated on a dataset from a different *target domain* (e.g. the literary domain), which is challenging due to the difference in data distribution. To alleviate this issue, we propose to use a data augmentation technique, *mention replacement* [58], which allows injecting names of a specific style in the training dataset.

This chapter is based on our work published in the COMHUM 2022 workshop [13] and the TALN conference [14]. It is organized as follows: we first introduce the issue of unseen name styles in more detail in Section 4.1. We mention related work on cross-domain NER and introduce mention replacement in Section 4.2. We describe the experiments through which we test the possible benefits of mention replacement in Section 4.3, and their results in Section 4.4. We perform these experiments in a cross-domain setup where the source domain is newswire articles and the target domain is fantasy literature. Given the high difference between these two domains, these experiments serve to check if our method can increase performance when source and target domains are distant. To see if the method is beneficial when these domains

are close, we perform additional experiments in Section 4.5 on a setup where literary data is available, but the target domain is of a specific literary genre (fantasy) not present in the training data. We release all the source code needed to reproduce our experiments under a free license[1].

4.1 UNSEEN NAME STYLES

Dekker et al. [59] perform a study where they evaluate several off-the-shelf NER systems for the purpose of extracting character networks from literary texts. They observe that NER performance varies greatly across novels, ranging from good (92.31 F1) to terrible (2.13 F1). While not as important, we also observe this variation when evaluating Renard in Section 3.4.3. Dekker et al. [59] find that a large part of the false negatives produced by the evaluated models (i.e. undetected mentions) stem from two specific types of names: word names (names that are also regular words, such as `Valor` or `Mercy`) and apostrophed names (such as `Rand al'Thor` or `Shai'Tan`). In the case of word names, they observe that replacing them with more common names (such as `Richard` or `Vincent`) results in better detection: in Glen Cook's fantasy novel *The Black Company*, where a lot of characters have a nickname formed by a common noun or an adjective (such as `Sleepy`), they observe a very large improvement from 6.85 to 90.00 F1-score. In the case of apostrophed names, simply removing the apostrophe yields performance improvements: in Alexandre Dumas' *The Three Musketeers*, removing the apostrophe from mentions of the main character `D'Artagnan` yields an improvement from 13.91 to 53.00 F1-score. Together, these experiments tend to show that the issue regarding these names is linked to their *form*, and not to their surrounding context.

We suspect that these shortcomings come from the datasets used to train the NER systems assessed by Dekker et al. [59]. Indeed, existing NER systems are usually trained on the main publicly available datasets (such as Ontonotes [213], CoNLL-2003 [195] or WikiGold [28]), which come from a few domains (news, Web...). Among these mainstream datasets, none contain texts of the literary domain, and annotating new datasets is costly. Applying off-the-shelf NER systems to literary texts likely suffers from these systems integrating knowledge specific only to their training data. This may lead to a mismatch in style between typical person names from training data and literary texts. This seems to be particularly the case in the fantasy genre, where character names can have very unusual styles depending on the setting.

4.2 CROSS-DOMAIN NER AND MENTION REPLACEMENT

The setting where the domain of the training data is different from the one of the testing data is called *cross-domain NER*, and a few techniques exist in the literature for that specific setting. Gururangan et al. [88] survey domain-adaptive pretraining, a technique where a pretrained model continues pretraining on the target domain, and show that it leads to performance gains. While it achieves gain on multiple domains, domain-adaptive pretraining is computationally expensive as a large amount of unlabeled data from the target domain is employed for pretraining. Ding et al. [61] train a generative language model to produce new examples by directly including NER tags in the generated text. Chen et al. [50] train a neural architecture to transform examples from the training domain to examples closer to the test domain. More recently, Yang et al. [231] propose *FactMix*, a two-step data augmentation process performing mention replacement followed by masked token replacement. However, as far as we

1 https://github.com/CompNet/ddaugNER

Augmentation List	Example Names
`conll`	*Pierre Fulke, C. Rangaranjan, Craig Stadler*
`wgold`	*John McLaughlin, Penley, Fresquito Fresquet*
`fantasy`	*Minbid-Dal, Thaminiel Stone-Singer, Jalrana Ostorius Halfmoon*
`novelties`	*Dancing, Rand al'Thor, Logen Ninefingers*

Table 6.: Example names sampled from different augmentation lists.

know, previous works are not specifically concerned with literary texts or the mismatch in person names between domains. In this chapter, we propose a first look into that problem, and propose to use a comparatively simpler technique based on data augmentation.

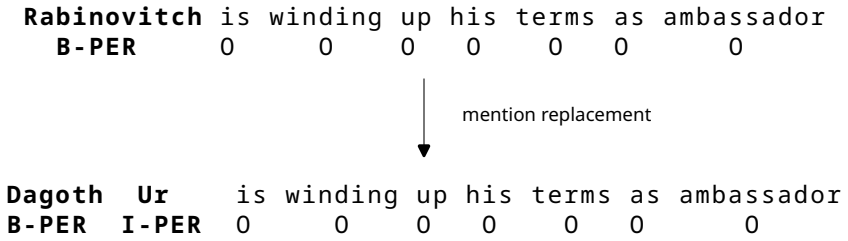


Figure 11.: Example of mention replacement in a source sentence from CoNLL-2003 [195]. We replace the mention `Rabinovitch` with the fantasy-style `Dagoth Ur`. In this specific example, the sentence labels have to be slightly modified to account for the higher number of tokens of the replacement entity.

Generally speaking, data augmentation is the generation of new synthetic training examples. Data augmentation techniques can be used to address data scarcity problems, or to increase the diversity of the training dataset to reduce overfitting. While these techniques are ubiquitous in image processing [181], they are less commonly explored in NLP [72], and even less for NER [58]. In an article from 2020, Dai and Adel [58] survey a few simple data augmentation schemes for NER, and show that they can improve NER performance, especially in the case of small training datasets. Among the augmentation scheme they survey, we are specifically interested in mention replacement, which can inject new names in the training dataset. It consists of generating new examples by replacing tokens from an entity mention found in the training dataset with tokens from another mention of the same type. Figure 11 shows an example of this process. Based on the same principle, we propose to randomly replace training mentions with ones from an *augmentation list* of names of the same domain as the test dataset.

4.3 EXPERIMENTS

4.3.1 *Mention Replacement*

To perform mention replacement, we compose lists of replacement names that do not come directly from the evaluation dataset. When generating a new example, we take an existing training example, copy it and replace one of its annotated mention by uniformly sampling from the augmentation list. In order to keep sentences as coherent as possible, if the same name originally appears in multiple mentions, we replace each of its occurrences by the same sampled name.

Name Set	Exact Match	Partial Match	Unseen
conll (train)	8.35%	67.46%	24.19%
wgold	3.58%	57.41%	34.92%
fantasy	10.39%	71.21%	18.40%
novelties	100%	0%	0%

Table 7.: Overlap of different augmentation lists with the set of character names from our evaluation dataset. The "exact match" column reports the proportion of PER labeled tokens from our evaluation dataset that are also in the studied name set. The "partial match" column is the proportion of PER labeled tokens containing a wordpiece [225] that exists in the studied name set. Finally, the "unseen" column indicates the rest of PER labeled tokens from the evaluation dataset.

In order to understand whether injecting names from the target domain is really beneficial, we compare different lists:

- **Internal Augmentation (`conll`)** We compose an augmentation list with all the PER forms of the training CoNLL-2003 dataset [195]. This augmentation list enables us to check the impact of our data augmentation scheme when no data from another domain is injected in the training dataset.
- **External Out-Of-Domain Augmentation (`wgold`)** We constitute an augmentation list with the PER forms of the WikiGold [28] NER dataset. WikiGold contains 145 annotated Wikipedia articles. Our goal with this list is to assess the impact of injecting new data in the training dataset, when data are outside the target domain.
- **External In-Domain Augmentation (`fantasy`)** We scrap the entirety of the names from Bethesda's *The Elder Scrolls* series of video games that are mentioned on the *Unofficial Elder Scrolls Pages*, a wiki dedicated to *The Elder Scrolls* universe[2]. In total, we retrieve 22,748 first names, 4,879 last names, 647 suffixes and 37 prefixes. We combine these to form new names: when performing mention replacement, we randomly compose a new name by first sampling from a set of valid forms (`[first name]+[last name]`, `[prefix]+[first name]+[last name]`, ...). We weight this sampling by a rough approximation of each form's frequency. When a form has been selected, we uniformly sample a name part for each of the form's elements.
- **Perfect Augmentation (`novelties`)** We create an augmentation list with all the annotated mentions of the test dataset. We use this list to establish the upper bound of our augmentation strategy.

Table 6 shows a few example names sampled from these lists. Meanwhile, Table 7 shows the overlap of our augmentation lists with the set of names from the fantasy subset of Novelties, our evaluation dataset. As expected, our `fantasy` list is closest to it, with 10% of Novelties PER tokens exactly matching in the list and 71.21% partially matching.

Since we are interested in knowing the impact of the number of generated examples, we compare the performance of models trained with different augmentation rates. We define the *augmentation rate* as the ratio of examples generated over the dataset size, so an augmentation rate of 1 would mean generating as many examples as there are examples in the dataset, and an augmentation rate of 0 corresponds to the vanilla dataset.

2 https://en.uesp.net/wiki/Main_Page

Galladon	Gandalf	Vin	Camon	Raoden
Pug	Selia	Chivalry	Arlen	Bran
Bilbo	Jon	Jake	Mercy	Laird
Kip	Silvy	Bug	Tam	Cotillion

Table 8.: Top 20 character names that were better detected by a model trained with data augmentation (10% augmentation rate) but not by a model trained with the original dataset only.

4.3.2 *Datasets*

TRAINING DATASET We train our model on a slightly modified version of CoNLL-2003 [195], which is one of the best known and used NER dataset. CoNLL-2003 is composed of 14,041 sentences from news articles. Our modifications consist of including honorifics as part of annotated mentions to be consistent with our evaluation dataset (see below). The training dataset contains annotations for four classes: persons (`PER`), organizations (`ORG`), locations (`LOC`) and miscellaneous (`MISC`). Despite restricting ourselves to character detection (`PER` class), we keep annotations for every class, since we observe that training the model with all classes increases NER performance.

EVALUATION DATASET We evaluate our NER model on the fantasy subset of our Novelties 0.1.0 dataset. This subset consists of approximately one chapter from 17 fantasy novels. We chose to focus on fantasy as we expect fantasy names to be challenging to detect given their particularities, as observed by Dekker et al. [59]. The testing dataset only contains `PER` mentions.

4.3.3 *Training and Evaluation*

We fine-tune a BERT model [60] with an added classification head on our training dataset, and report its performance on our evaluation dataset depending on the augmentation rate. In order to obtain stable results, we report the mean of the results from 10 fine-tuning runs for each augmentation rate that we test. We train models without augmentations for two full epochs, while we adjust the number of training steps for models with augmentation so that they learn on the same number of examples as models without augmentation. We use a learning rate of $2 \cdot 10^{-5}$ [60], and we initialize each model with the `bert-base-cased` checkpoint of the *transformers* library [223]. At inference time, each model predicts NER tags separately for each sentence, and we do not provide it with any additional context. Since the evaluation dataset only contains annotations for the `PER` class, we simply discard predictions made for other classes.

4.4 RESULTS

Results can be seen in Figure 12. We report performance according to the CoNLL-2003 guidelines [195] and using the Python `seqeval` library [144]. We observe an increase in recall and a decrease in precision for all augmentation lists.

Injecting in-domain names greatly increases recall, as can be seen with the results of using the `fantasy` and `novelties` augmentation lists. Our `fantasy` augmentation scheme increases recall by up to roughly 7.5 points, at the cost of some precision. Table 8 shows the top-20

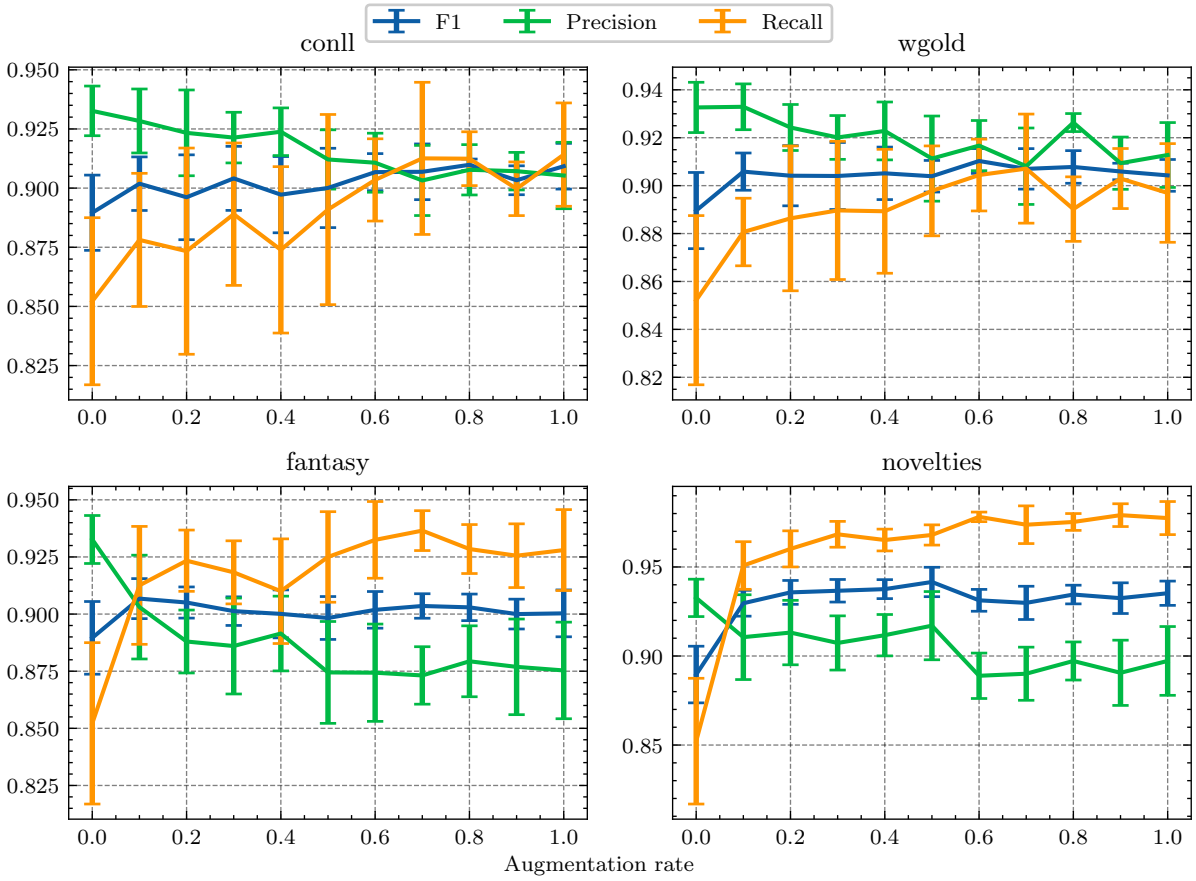


Figure 12.: Precision, recall and F1-score against augmentation rate for different augmentation lists. Each data point corresponds to the mean of 10 runs. Error bars represent the 95% confidence interval.

names that were better recalled using the `fantasy` augmentation. These are mostly fantasy names that do not appear in the CoNLL-2003 training dataset. Adding perfect information increases recall by up to 11 points, showing that large gains are possible. Meanwhile, injecting out-of-domain names in the training dataset with the `conll` or `wgold` lists trade recall, but in a smaller proportion. This shows the importance of injecting data from the target domain to improve performance.

To try to understand the decrease in precision when using data augmentation, we perform some additional experiments. Based on our manual investigation of the false positives, we formulate two hypotheses that we want to test:

h_1 Adding more examples with `PER` mentions modifies the class distribution in the dataset, progressively leading to a situation of class imbalance where these mentions are way more frequent than those of the other classes (`LOC`, `ORG` and `MISC`). As we observe that removing other classes from the training dataset is detrimental to `PER` detection performance (possibly because having more classes makes the model better at distinguishing between them), we suppose that class imbalance can be detrimental as well.

h_2 Increasing name variety in the training set encourages the model to try and make `PER` predictions for tokens whose classes are ambiguous given a specific sentence. We deem a token "ambiguous" when the context given to the model does not suffice to infer its correct class (for example, when it is not possible to distinguish between a `PER` and a `LOC` mention using only an input sentence).

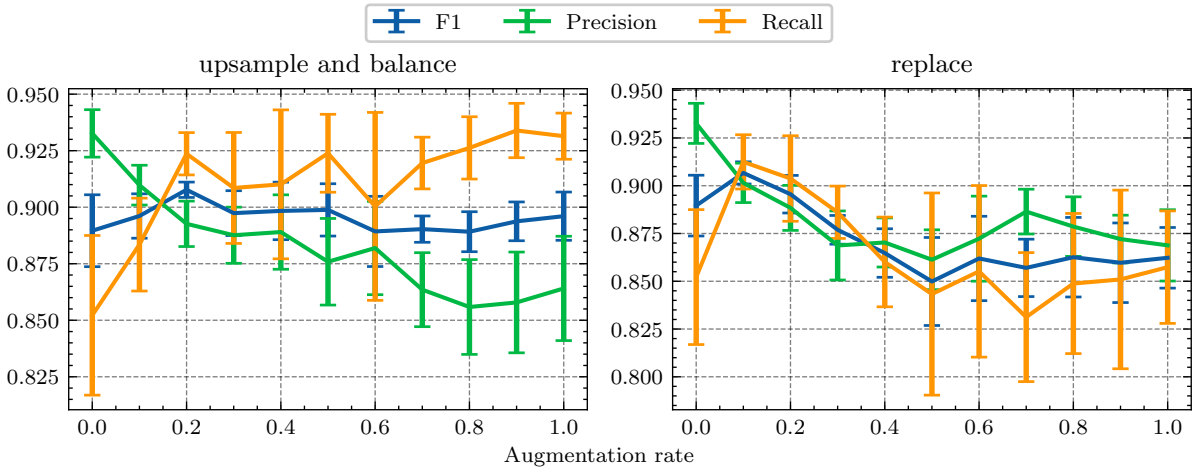


Figure 13.: Precision, recall and F1-score against augmentation rate for different augmentation methods, using the `fantasy` augmentation list. Error bars represent the 95% confidence interval.

4.4.1 *Class Imbalance*

To check if the decrease in precision is due to class imbalance, we test two different methods to inject names in the training dataset that should not suffer from the class imbalance problem:

- `Upsample and balance`: After adding newly generated examples to the dataset, we copy some of the original examples and add them to the dataset to restore the original class distribution.
- `Replace`: Instead of *adding* newly generated examples to the dataset, we *substitute* them for the original training samples.

Figure 13 shows the performance of a model trained on a dataset modified with the above augmentation methods. The *upsample and balance* strategy still increases recall, but does not fix the precision issue. Meanwhile, the `replace` strategy still makes precision decrease, and also generally decreases performance for an augmentation rate greater than 20%. As none of these two methods are able to fix the precision issue, we conclude against h_1: class imbalance is not a plausible cause of precision decreasing.

4.4.2 *Name Variety and Ambiguous Occurrences*

In order to check h_2, we analyze the difference in false positives between a model trained with augmentation and a model trained without it. As errors can vary between runs, we perform three training runs for each model and keep the false positives that are consistent between the runs. We then observe the set of false positives of the model trained with augmentation that are not present in the set of false positives of the model trained without augmentation. This allows us to analyze errors that are specific to our augmentation scheme.

We manually find that for 51% (41/81) of these false positives, the model attributes a `PER` label to some tokens even though the context alone is not sufficient to know the correct label of the mention, such as in the following example from *Elantris*:

```
Raoden stood, and as he did, his eyes fell on Elantris again.
```

Here, the model correctly predicts that `Elantris` is a mention to detect, but gives it the class `PER` while `Elantris` is actually a city (which corresponds to the `LOC` class). Of course,

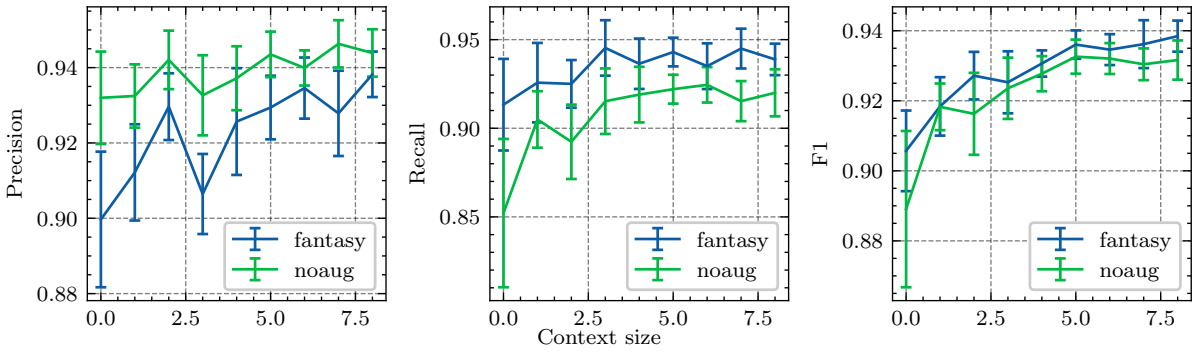


Figure 14.: F1-score, precision and recall against different context sizes for a model trained without augmentation (`noaug`) and a model trained with augmentation (`fantasy`) (60% augmentation ratio). Error bars represent the 95% confidence interval.

predicting the correct class of `Elantris` in this example would only be a matter of luck, as the given context alone is not sufficient.

Overall, this result shows a bias towards the `PER` class, that was introduced by increasing the variety of possible names at training time. To try and mitigate this effect, we supply the model with a broader context, in the hope that this context can help to resolve some ambiguities. In our previous example, the next sentence gives enough context to predict the correct entity type for `Elantris`:

```
Raoden stood, and as he did, his eyes fell on Elantris again.  Resting
in the great city's shadow, Kae seemed like an insignificant village by
comparison.
```

We define a context size of n as giving the model the n sentences that *precede* the considered sentence, as well as the n sentences that *follow* it.

Figure 14 shows the effect of giving more context to models trained with augmentation (`fantasy`) or without it (`noaug`). As can be seen, context is beneficial to both models, steadily increasing F1-score. Precision increases a lot for the model trained with augmentation, as previously ambiguous mentions are now correctly labeled. Using our previously described method for error analysis, we find that only 40% (33/82) of the mentions found in false positives are deemed ambiguous for a context size of 1. Meanwhile, the recall augmentation for the `noaug` model can be attributed to it predicting `PER` mentions in previously ambiguous cases. While adding more context results in greater recall for the `noaug` model, it still never surpasses the recall of the `fantasy` model.

4.5 Adaptation from literary texts to a specific genre

Our experiments show that, in the case of cross-domain NER, injecting names of the target domain into the training dataset can improve recall at the cost of some precision. However, we performed our previous experiments in a setup where the source domain (news) is largely different from the target domain (fantasy). In a more practical setup, researchers might have access to a few texts from one or several literary genres (source domain), and want to improve performance on an unseen genre (in our case, fantasy). To understand whether our augmentation scheme is beneficial for this use case, we perform an additional experiment: we repeat the same experiment as above, but we change the training dataset to the non-fantasy portion of Novelties v0.1.0 (excerpts from the 23 remaining novels). We keep the fantasy subset of

Novelties v0.1.0 as the test set. We supply the model with the largest context size we survey in previous experiments (8), as it results in the highest performance.

4.5.1 *Results*

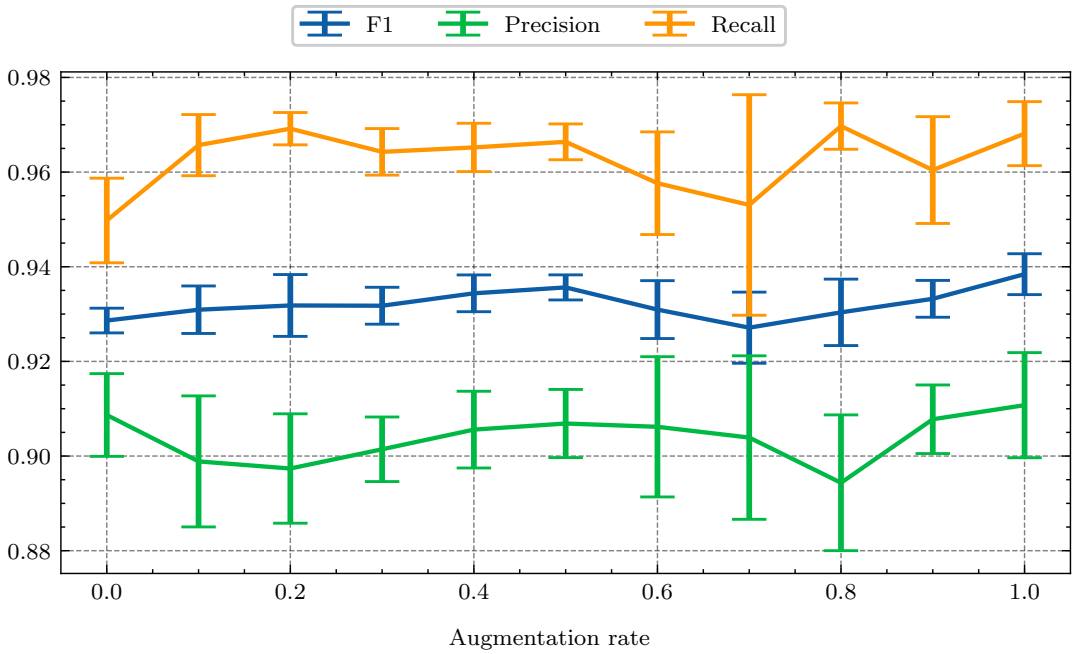


Figure 15.: F1-score, Precision and Recall against augmentation rate for a model trained on the non-fantasy portion of Novelties v0.1.0. We perform augmentation with the "fantasy" list.

The base performance when training on a literary corpus is, as we can expect given the proximity between the training and test domains, higher than when training the model on the CoNLL-2003 corpus. Compared to that setup, we also observe a general increase in recall and decrease in precision, albeit in smaller proportions.

4.6 CONCLUSION

In this chapter, we demonstrated the usage of a simple data augmentation technique to better detect fantasy names when performing NER. This technique is both simple to implement and computationally cheap when compared to other techniques such as domain-adaptive pretraining [88]. While this technique greatly increases model recall, precision decreases as the model sometimes does not have enough information to predict the correct entity class of some mentions. We showed that this issue can be mitigated by giving the model more local information. However, for the same context size, our technique always results in greater recall but lower precision. In the context of character detection, we argue that this trade is beneficial: while false positives can be filtered after performing NER (automatically or manually), false negatives are not easily recovered. We also note that increasing the NER context has a positive impact on recall in general.

In this chapter, we focused on local neighboring context, and we show that it helps disambiguate some mentions. Nevertheless, the information necessary to disambiguation is not always present near the processed sentence, and might exist in more remote parts of the novel.

In the next chapter, we further investigate the role of context in NER, and we question the benefits of document-level context when it comes to entity type disambiguation.

5

CONTEXT RETRIEVAL TO ALLEVIATE TRANSFORMERS RANGE LIMITATION

Contents

5.1 Related Work 54
5.2 Method 55
5.2.1 Document-Level Context Retrieval 56
5.2.2 Context Retrieval Dataset Generation 56
5.3 Experiments 58
5.3.1 Unsupervised Retrieval Baselines 58
5.3.2 Inference 58
5.3.3 Re-ranker Baselines 59
5.3.4 Re-Ranker Training 59
5.3.5 Named Entity Recognition Model 60
5.3.6 Evaluation 60
5.4 Results 60
5.4.1 Neural Re-Ranker Configuration Selection 60
5.4.2 Comparison with Unsupervised Retrieval Methods 61
5.4.3 Comparison with Re-Rankers 62
5.4.4 Size of the Context Window 62
5.5 Conclusion 64

We show in Chapter 3 that the performance of NER has a fundamental impact on character network extraction. While pre-trained transformer-based models are able to solve the NER task with a very high F1-score [60, 229], the quadratic complexity of their attention mechanism means taking an entire long document as input is computationally costly [192]. The common way of avoiding this pitfall is to process documents block by block, in a rolling window fashion. In that case, we our model suffers from a *range limitation*: it is unable to obtain information from the whole document. Yet, document-level context is useful for entity type disambiguation: for NER, lacking access to this context can result in errors [188]. This can occur when the type of an entity mention cannot be inferred from the local context. For instance, in the following sentence from the fantasy novel *Elantris*, one cannot decide if the underlying entity of mention `Elantris` is a person (`PER`) or a location (`LOC`) without prior knowledge:

> *"Raoden stood, and as he did, his eyes fell on Elantris again."*

In the novel, this prior knowledge comes from the fact that a human reader can recall previous mentions of `Elantris`, even at a very long range. A vanilla transformer-based model applied sequentially with a rolling window, however, might make an error without a neighboring sentence clearly establishing the status of `Elantris` as a city.

To try and solve the range limitation of transformers, a number of authors recently explored "efficient transformers", with a sub-quadratic complexity (see Tay et al. [193] for a survey). Still, long sequences remain challenging [192]. We posit that a model does not need

the entire document to successfully carry NER, but that a limited number of relevant context spans is enough. In that case, retrieval-based methods can circumvent these issues, by retrieving relevant context from the document and concatenating it to the input. Unfortunately, while some general context retrieval datasets such as MSMarco [27] exist, no context retrieval dataset is available for NER, and manually annotating such a dataset would be costly. This lack of data prevents from using a specialized supervised context retrieval method.

Meanwhile, instruction learning has very recently seen an increasing focus from the NLP community [165]. Instructions-tuned models are able to solve a variety of tasks (classification, regression, recommendation...) provided they can be formulated as text generation problems. Inspired by the progress of these models, we propose to generate a synthetic context retrieval dataset using an LLM in order to train a supervised retrieval model. Since applying this retriever on a whole document is costly, we instead use it as a re-ranker, by first retrieving a set of candidate contexts using simple retrieval heuristics and then keeping the best sentences among these.

In this chapter, based on our work published in the ACL 2023 [17] and EMNLP 2023 [16] conference, we make the following contributions:

- We develop a method to generate a synthetic context retrieval dataset adapted to the NER task.
- We train a neural context retriever on that dataset, and use it at NER inference time to retrieve relevant context.
- Through comparisons with existing unsupervised and supervised retrievers, we highlight the benefits of our re-ranker for NER performance.

We release our code and data to reproduce our experiments under a free license[1].

We organize this chapter as follows. First, we quickly survey related works in Section 5.1. We then describe the context retrieval method we propose in Section 5.2, including how to generate a synthetic context retrieval dataset using an LLM and the design of our neural context retriever. We describe the experiments we perform to test this retriever in Section 5.3, and analyze their results in Section 5.4.

5.1 Related work

NER and context retrieval Different *external* context retrieval techniques have been leveraged for NER [123, 209, 240]. External resources in which to retrieve context can include online wikis or knowledge bases. Comparatively, few studies focus on document-level context retrieval, which can be used even when no external resources are available. While popular texts have available resources, this lack of resource is realistic for lesser known ones. Related to our work, Luoma and Pyysalo [125] introduce majority voting to combine predictions made with different contexts, but their study is restricted to neighboring sentences and ignores document-level context. Comparatively, we propose a retrieval system based on retrieval and re-ranking at the document level.

Re-rankers In the field of information retrieval, re-rankers are used to complement more traditional retrievers such as BM25 [166]. In that setting, given a query and a set of passages, a traditional retriever is first used to fetch a certain number of candidate passages. Then, a more computationally expensive re-ranker such as MonoBERT [147] or MonoT5 [148] refines

1 https://github.com/CompNet/conivel

that batch by retrieving the top-k passages among these candidates. While supervised re-rankers offer good performance, they necessitate training data, and annotation can be costly. To avoid this issue, we propose a re-ranker tailored to the NER task that is trained on a synthetic retrieval dataset generated using an instruction following LLM.

INSTRUCTION-FOLLOWING LLMS Instruction-following LLMs are trained to output text by following user instructions. Multitask learning [139, 177, 211] and fine-tuning on instructions datasets [51, 191] are two common training paradigms for these models. One of the goals of these training methods is to obtain good zero-shot performance for a variety of tasks, making instruction-following models very versatile. Our synthetic dataset generation task calls for models producing longer outputs than multitask learning-based models such as BloomZ or mTo, which are biased in favor of short outputs [139]. We therefore select Alpaca [191], an instruction-tuned model based on Llama [197].

5.2 METHOD

In this section, we start by introducing the *document-level context retrieval* task for NER in Section 5.2.1, as well as the context retrieval model we propose to use. Then, since no dataset exists for this task, we detail how we generate a synthetic context retrieval dataset using an instruction-following LLM in Section 5.2.2.

Description (all classes)	**Prompt template** `'{INPUT SENTENCE}' - In the preceding sentence, {ENTITY MENTION} is a character. Invent a one-sentence description for this character, mentioning their name.` **Example input sentence** *[One-Eye]'s handicap in no way impairs his marvelous hindsight.* **Example generated context** *One-Eye is a wise and mysterious character with a penchant for coming up with invaluable insights after the fact.*
Action (`PER` only)	**Prompt template** `Invent a single sentence depicting the character '{ENTITY MENTION}' performing an action, mentioning their name.` **Example input sentence** *"It's my stomach, [Croaker],"* **Example generated context** *Croaker was whistling a jaunty tune as he strolled through the Park.*
Movement (`LOC` only)	**Prompt template** `Invent a single sentence depicting a character of your invention going to {ENTITY MENTION}. You must mention the name of the character.` **Example input sentence** *Lightning from a clear sky smote the [Necropolitan Hill].* **Example generated context** *The gothic vampire Count Necropolis ventured to Necropolitan Hill, his ancient stomping grounds.*

Table 9.: Prompts templates and examples of positive context retrieval samples generated by Alpaca-7b.

Positive examples swapping	**Example input sentence** *We left in pretty good time and came after nightfall to Klausenburgh* **Example context** *Forley was an adventurous and daring individual who was never afraid to take risks*
Negative sampling	**Example input sentence** *I am afraid that I have been tempted into too great length about the Italian Catherine; but in truth she has been my favourite.* **Example context** *said Alice, as she swam about, trying to find her way out.*

Table 10.: Examples of negative context retrieval samples generated by *positive examples swapping* or *negative sampling*.

5.2.1 *Document-Level Context Retrieval*

We define the document-level context retrieval problem as follows: given a sentence s_i and its enclosing document D_i, we must retrieve S_i, a set of k relevant sentences in D_i. We define relevance as being helpful for predicting the entity mentions in s_i by allowing entity class disambiguation. After retrieving S_i, we concatenate its sentences to s_i, in order to form a list $\hat{S}_i$, while keeping the relative ordering of sentences in D_i. After context retrieval, we compute NER labels for sentence s_i by inputting $\hat{S}_i$ into a NER model and keeping only the labels related to s_i. Figure 16 provides an overview of our retrieval process.

In practice, for our neural context retriever, we use a BERT model [60] followed by a regression head, which is reminiscent of MonoBERT [147]. For a given sentence and candidate context, we concatenate them using a special `[CLS]` token and input the resulting text into our model. The retriever outputs the estimated relevance of the candidate context for the input sentence between 0 and 1. Because applying our model on the whole document is computationally costly, we use our retriever as a re-ranker: we first retrieve some *candidate contexts* using cheaper heuristics such as BM25 [166] (cf. Section 5.3.2) before re-ranking these sentences.

5.2.2 *Context Retrieval Dataset Generation*

No context retrieval dataset exists for the task of retrieving relevant context for NER at the document level. Without supervision, we cannot train a supervised model to solve this task and are restricted to unsupervised retrieval methods. Therefore, we set out to generate a synthetic dataset using the instructions-tuned LLM Alpaca [191].

We define a context retrieval dataset as a set of 3-tuples (s_i, s_j, y), where:

- s_i is the input sentence (the sentence for which we wish to retrieve context).
- s_j is the retrieved context sentence.
- y is the relevance of s_j with respect to s_i: either 1 if s_j is relevant, or 0 otherwise.

Depending on whether the example is positive ($y = 1$) or negative ($y = 0$), we design different generation methods.

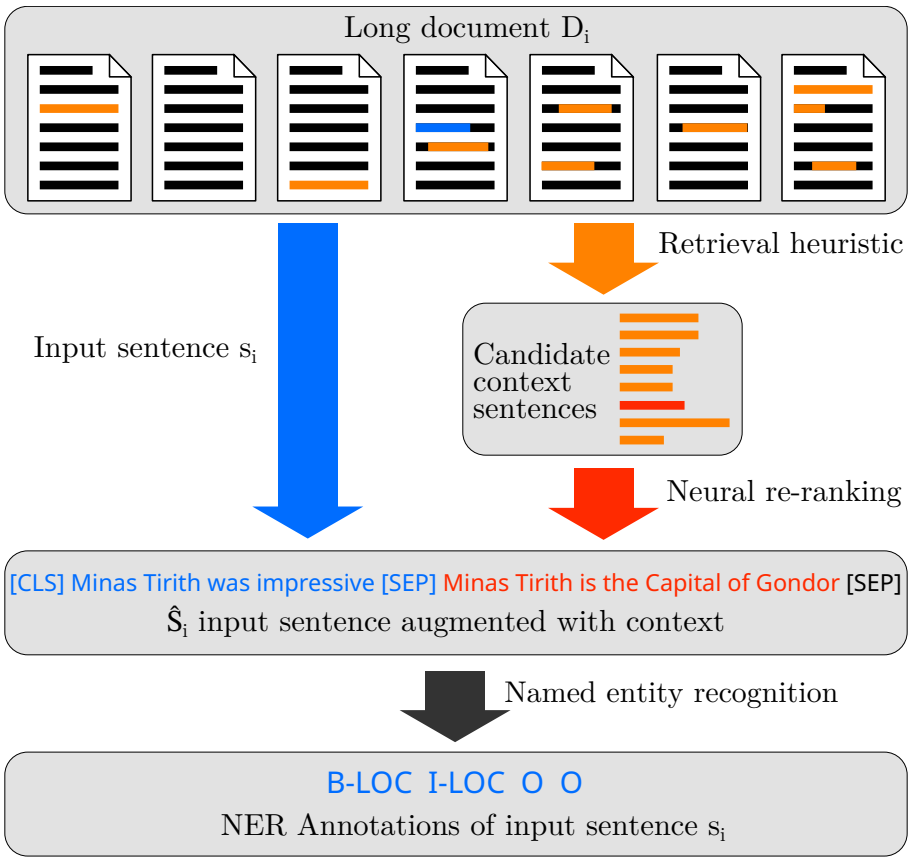


Figure 16.: Overview of our neural re-ranker performing context retrieval.

POSITIVE EXAMPLES For each NER class in our dataset (PER (person), LOC (location) and ORG (organization)), we empirically observe which types of sentences can help for disambiguation with other classes. We determine that the following categories of sentences are useful:

- For all classes (PER, LOC and ORG): descriptions of the entity explicitly mentioning it (**Description**).
- For the PER class only: sentences describing the PER entity performing an action (**Action**).
- For the LOC class only: sentences describing a PER entity going to the LOC entity (**Movement**).

We design a prompt for each of these types of sentences that, given an input sentence and an entity mention, can instruct Alpaca to generate a sentence of this type. Table 9 shows these prompts and some example sentences generated by Alpaca-7b. Given a sentence s_i from our NER training dataset, we uniformly sample an entity mention from s_i and generate s_j by instructing Alpaca with one of our prompts. This allows us to obtain a positive context retrieval example $(s_i, s_j, 1)$. We repeat this process on the whole NER training dataset. To avoid overfitting on the most frequent entity mentions, we generate exactly one example per entity form in the dataset. We filter out sentences s_j that do not contain the target entity mention string from the input sentence s_i.

NEGATIVE EXAMPLES We use two different techniques to generate negative examples:

- *Negative sampling*: given a sentence s_i from a document D_i, we sample an irrelevant sentence by randomly selecting a sentence from another document D_j. Negative sampling

generates some contexts with entity mentions, but most of them do not contain any entity mention.

- *Positive examples swapping*: given an existing positive example $(s_i, s_j, 1)$, we generate a negative example $(s_i, s_k, 0)$ by replacing s_j with a context s_k from another positive example. Since we generate a single positive example per entity form, s_k has a high chance not to contain an entity mention from s_i. All context sentences s_k contain an entity mention. We use this additional generation technique because we found that generating examples using only negative sampling led to the model overfitting on contexts containing entity mentions (since negative sampling does not guarantee that the retrieved context contains an entity mention).

GENERATED DATASET We generate two context retrieval datasets: one with Alpaca-7b, and one with Alpaca-13b. They contain respectively 2,716 and 2,722 examples[2]. Interestingly, the models display a knowledge of some entities from the dataset when generating samples, probably thanks to their pre-training. For example, when asked to generate a context sentence regarding *Gandalf the Grey* from *The Lords of the Rings*, an Alpaca model incorporates the fact that he is a wizard in the generated context even though it does not have this information from the input text.

5.3 EXPERIMENTS

5.3.1 *Unsupervised Retrieval Baselines*

To see if training our supervised context retriever is beneficial for NER, we compare its performance with the following unsupervised baselines:

- `no retrieval`: The model performs the NER task on each sentence separately, without additional context.
- `surrounding`: Retrieves sentences that are just before and after the input sentence. This baseline is equivalent to a NER model applied sequentially, where the model only has access to local context as opposed to global, document-level context.
- `bm25`: Retrieves the k most similar sentences according to the BM25 scoring scheme [166]. BM25 computes the similarity between two pieces of texts by considering their common terms and their relevance.
- `samenoun`: Retrieves k sentences that contain at least a noun in common with the input sentence. Our hypothesis is that sentences with common nouns will likely contain common entity mentions, which might help the model to disambiguate the class of input mentions. We identify nouns using the NLTK [38] library.

5.3.2 *Inference*

At inference time, we retrieve context in the whole novel to ensure that the model has access to document-level context. To avoid heavy computational costs, we use our neural context

2 The number of examples between the two datasets is different because of the filtering process when generating positive examples. This filtering occurred very rarely, as the vast majority of generated sentences included the target entity mention.

retriever as a re-ranker. For each sentence s_i of the dataset, we first retrieve a total of $4n$ *candidate context sentences* using the following heuristics:

- n sentences using the `bm25` heuristic.
- n sentences using the `samenoun` heuristic.
- The n sentences that are just before the current sentence.
- The n sentences that are just after the current sentence.

With this setting, the same candidate sentence can be retrieved by different heuristics: to avoid redundancy, we filter out repeated candidates. We experiment with the following values of n: $\{4, 8, 12, 16, 24\}$.

After retrieving $4n$ candidate contexts, we compute their estimated relevance with respect to the input sentence using our neural context retriever. We then concatenate the top-k contexts to the input sentence s_i before performing NER prediction. We report results for a *number of retrieved sentences* k from 1 to 8.[3]

5.3.3 *Re-ranker Baselines*

We compare our system with the following trivial re-ranker baselines:

- `random re-ranker`: We retrieve $4n$ sentences with all the unsupervised heuristics (as described in Section 5.3.2), and then randomly select k sentences from these candidates.
- `bucket random re-ranker`: Same as above, except we select $k/4$ sentences for each heuristic.

These baselines allow us to observe the real benefits of re-rankers. We also compare our retriever with supervised re-rankers trained on the MSMarco passage retrieval dataset [27]:

- `bm25+monobert`: We retrieve 16 sentences using BM25, and then use the MonoBERT-large [147] re-ranker to retrieve the k most relevant sentences. This configuration simulates a classic information retrieval setup.
- `all+monobert`: We retrieve $4n$ sentences using all the unsupervised heuristics (as described in Section 5.3.2), and then use the MonoBERT-large re-ranker to retrieve the k most relevant sentences. This configuration is the closest to our neural retriever, so we use it to compare the influence of the training dataset.
- `bm25+monot5`: Same as `bm25+monobert`, but using a MonoT5 re-ranker [148].
- `all+monot5`: Same as `all+monobert`, but using a MonoT5 re-ranker.

5.3.4 *Re-Ranker Training*

We train our neural context retriever on the synthetic dataset we generated as described in Section 5.2.2 for 3 epochs with a learning rate of 2×10^{-5}. To study the importance of the LLM size that we use when generating our synthetic dataset, we generate our context retrieval dataset with two differently sized versions of the same model; as detailed in Section 5.2.2: Alpaca-7B (7 billion parameters) and Alpaca-13B (13 billion parameters) [191].

3 The `samenoun` heuristic is an exception, as it may retrieve fewer than k sentences in some cases. This is because some input sentences have fewer than k contexts in the document with at least one common noun.

5.3.5 *Named Entity Recognition Model*

In all of our experiments, we use a pre-trained BERT model [60] followed by a classification head fine-tuned for 2 epochs with a learning rate of 2×10^{-5}. We use the `bert-base-cased` checkpoint from the `huggingface transformers` library [223].

5.3.6 *Evaluation*

We evaluate models on the Novelties v0.1.0 literary NER dataset we introduce in Section 3.2.1. As a reminder, it is composed of the first chapter of 40 novels, and has 3 NER classes: Person (`PER`), Location (`LOC`) and Organization (`ORG`). We split each document into sentences. To perform context retrieval on the entire novels, we also collect the full text of each novel.

We compute the F1-score of our different configurations using the default mode of the `seqeval` [144] library for reproducibility. We perform all experiments 5-folds, and report the mean of each metric on the 5 folds. To compare the different retrieval configurations fairly, we train a single NER model and compare results using this model and different retrieval methods. For retrieval methods that are not deterministic (some re-rankers and the `samenoun` heuristic[4]), we report the mean of 3 runs to account for variations between runs.

5.4 RESULTS

In this section, we discuss the results of our experiments. We start in Section 5.4.1 by finding the optimal configuration of neural context retriever by surveying the number of candidate contexts to retrieve before re-ranking and the size of our Alpaca model. Then, we compare our retriever with unsupervised retrievers in Section 5.4.2, and with supervised re-rankers in Section 5.4.3. Finally, we study the influence of the context window (i.e. the window in which the retrieval system is allowed to retrieve context) in Section 5.4.4.

5.4.1 *Neural Re-Ranker Configuration Selection*

We first set out to find the optimal configuration of our neural context retriever. We survey the number of candidate contexts $4n$ to retrieve before re-ranking, and the size of the model used to generate the synthetic context dataset. Figure 17 shows the NER performance of the neural retriever trained with either Alpaca-7b or Alpaca-13b for different values of n.

NUMBER OF CANDIDATE CONTEXTS We find that $n = 4$ is too low, probably because not enough relevant candidate contexts are retrieved to obtain good performance. Increasing n leads to better performance for high values of the number of retrieved sentences k. However, the best performance is obtained for values of k between 2 and 5.

LLM MODEL SIZE We find that the size of the Alpaca model (7b versus 13b) has very little influence on the final NER performance for all surveyed values of n. Qualitatively, we observe that examples generated using Alpaca-7b are linguistically correct and respect our instructions, trumping the need for a larger model.

In the rest of this chapter, we discuss the best configuration we found in this experiment, unless specified otherwise: the neural retriever trained using Alpaca-7b and with $n = 8$

4 The `samenoun` method is not deterministic since it randomly retrieves among candidate sentences that have a common noun with the input sentence.

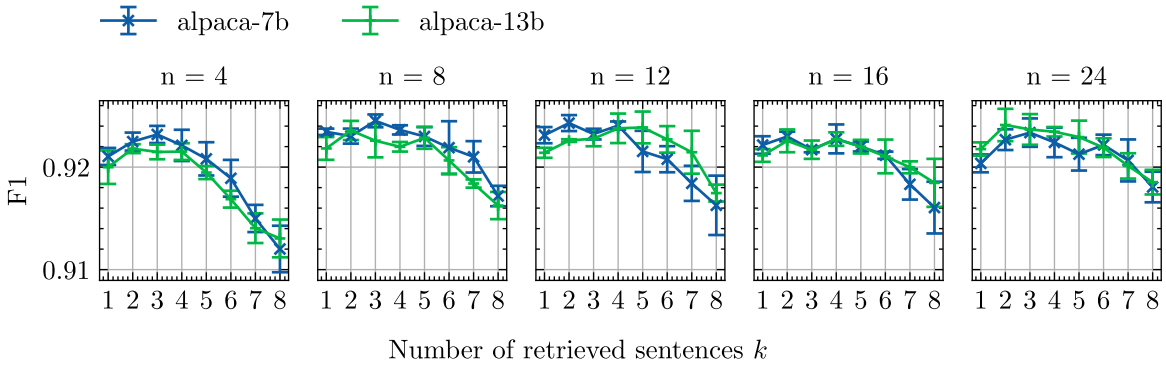


Figure 17.: Effect of the number of candidate sentences $4n$ on the performance of our neural context retriever, trained using a dataset generated with Alpaca-7b or Alpaca-13b.

and $k = 3$. We also keep the same configuration for the supervised baselines. With that configuration, our neural re-ranker achieves an F1-score of 98.01 on the evaluation portion of our synthetic context retrieval dataset.

5.4.2 *Comparison with Unsupervised Retrieval Methods*

Figure 18 shows the Precision, Recall and F1-score of our NER model when using our re-ranker or different unsupervised retrieval methods. Retrieval with any method is beneficial (except when retrieving 6 sentences or more with `surrounding` or `bm25`), but our neural re-ranker generally outperforms all the simple heuristics. Specifically, it beats the `no retrieval` configuration by around 1 F1 point, with its peak performance being when the number of retrieved sentences $k = 3$. All configurations benefits Recall (except `surrounding` when $k = 8$), but they can all have a negative impact on Precision depending on the value of k.

Input sentence	**Candidate contexts**	**Score**
"The Ministry of Truth – Minitrue, in Newspeak [Newspeak was the official language of Oceania."	*"The Ministry of Truth, which concerned itself with news, entertainment, education, and the fine arts."*	1.0
	"Winston made for the stairs."	0.0
"The eldest of these, and Bilbo's favourite, was young Frodo Baggins."	*"Then he disappeared inside with Bilbo, and the door was shut."*	1.0
	I feel I need a holiday, a very long holiday, as I have told you before.	0.0

Table 11.: Example predictions of the neural context retriever.

PER-BOOK RESULTS Figure 19 shows the F1 score per book for our retrieval method and the unsupervised retrieval methods, for a single retrieved sentence. Our neural context retriever is better or on par with other configurations for 50% of the novels (20 out of 40). Performance varies a lot depending on the novel, with the highest improvement of the neural method over the `no retrieval` configuration being 8.1 F1 points for *The Black Company*.

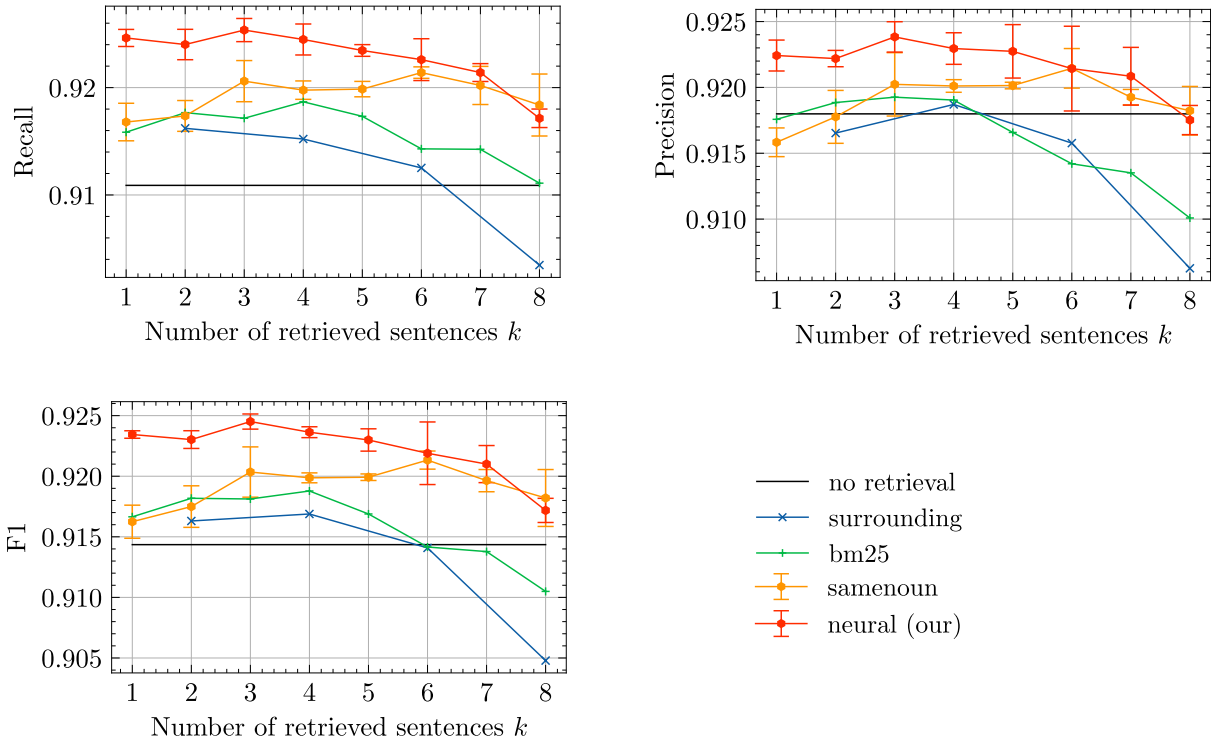


Figure 18.: NER Precision, Recall and F1-score comparison between our neural retriever and unsupervised retrieval methods.

5.4.3 *Comparison with Re-Rankers*

We compare the F1 of our neural re-ranker with existing re-rankers trained on MSMarco [27] in Figure 20. All the supervised methods outperform the random re-rankers, showing that they can indeed retrieve useful context. Our model outperforms the other supervised methods, albeit slightly. While results are close, this is still interesting as we used generated data only while the other supervised approaches were trained with gold annotated data. We note that using multiple pre-retrieval heuristics (the `all+` configurations and our neural re-ranker) is generally better than using only BM25, which tends to show that retrieving sentences with BM25 alone may lead to missing important context sentences.

5.4.4 *Size of the Context Window*

To understand the role of the size of the *context window*, we compare between retrieving context in the first chapter of each book only as in [17] and retrieving context on the entire book. Figure 21 shows the performance of the `bm25`, `samenoun` and `neural` configurations depending on the context window. For the `neural` configuration, since the number of retrieved sentences $4n$ may have an influence on the results, we report performance with values of n in $\{4, 8, 12\}$.

BM25 Retrieving a few sentences from the current chapter seems more beneficial to the `bm25` heuristic. However, performance decreases sharply when retrieving more than 2 sentences in the same chapter. This effect is weaker when retrieving context in the whole book, which seems indicative of a saturation issue, where the number of helpful sentences that can be retrieved using `bm25` in a single chapter is low.

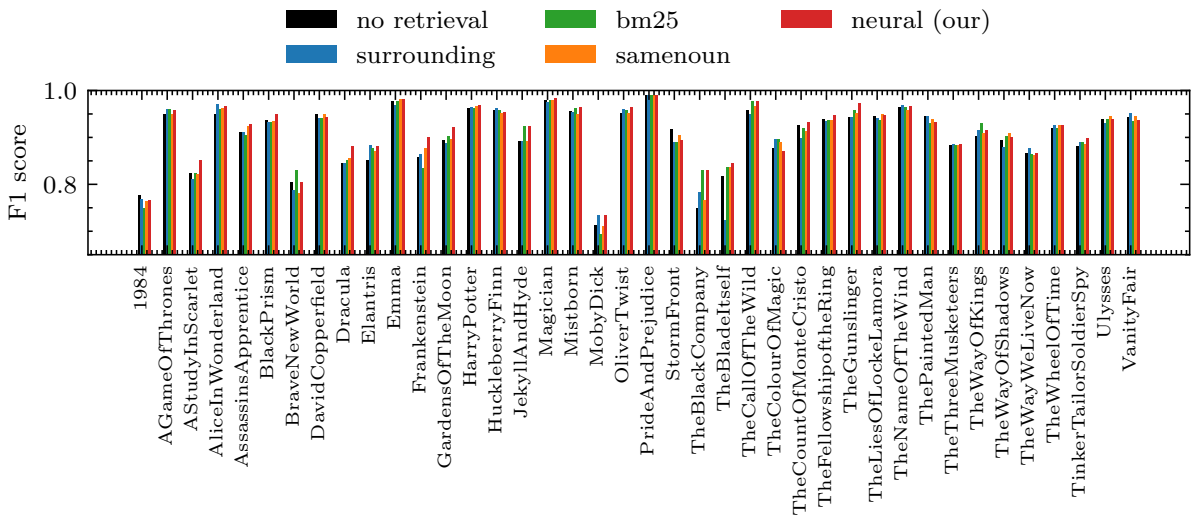


Figure 19.: NER F1 Score per book for different retrieval methods, with a number of retrieved sentences $k = 1$.

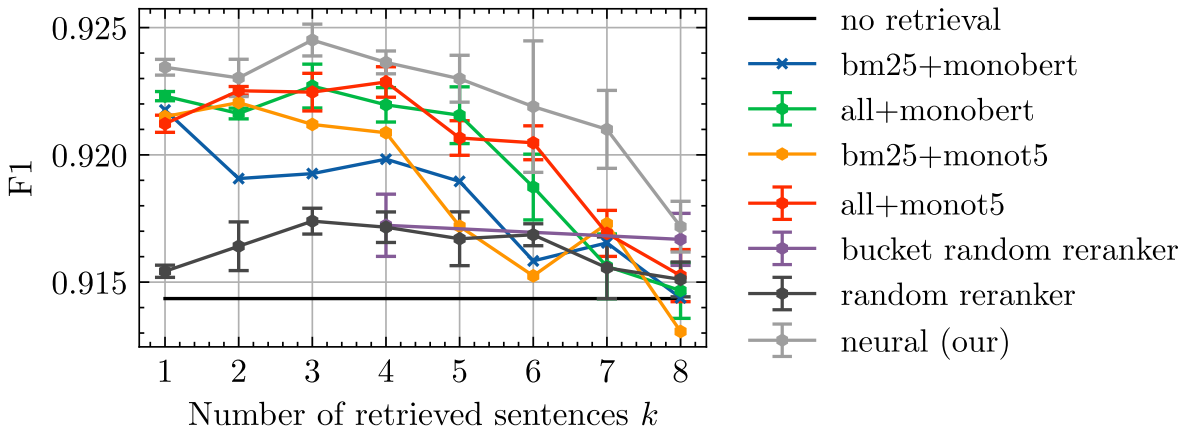


Figure 20.: NER F1 score comparison between our retriever and different re-ranking methods.

SAMENOUN The performance of the `samenoun` heuristic seems to follow the same pattern as `bm25`. Retrieving a few sentences from the current chapter seems better than doing so in the full document. This might be because the current chapter has a higher chance of containing sentences talking about an entity of the input sentence. Meanwhile, it is better to retrieve contexts in the whole book for values of $k > 4$, possibly because such a number of retrieved sentences means the heuristic has a high enough chance of retrieving a relevant sentence at the book level.

NEURAL The context window seems critical for the performance of the neural retriever. Retrieving context only in the current chapter suffers from a large performance drop compared to retrieving context in the whole book. This is true even for different values of the number of retrieved sentences $4n$.

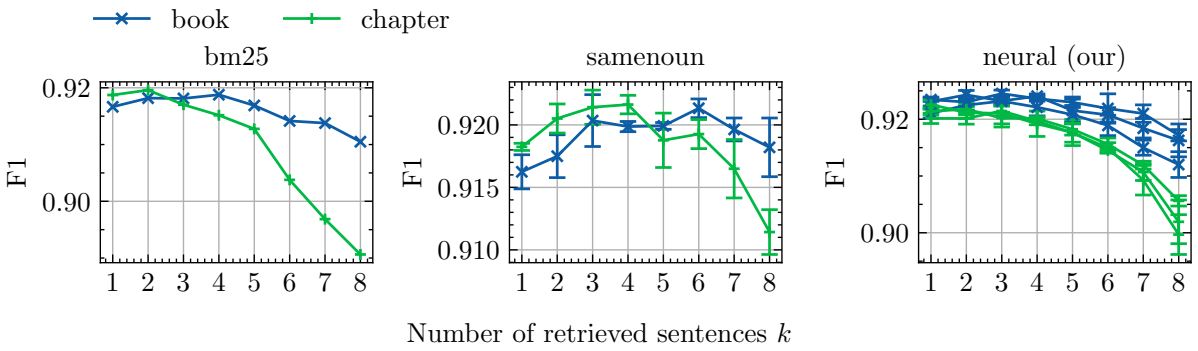


Figure 21.: NER performance of different retrieval methods for different context window sizes as a function of the number of retrieved sentences k. For the `neural` configuration, we report performance for values of n in $\{4, 8, 12\}$.

5.5 CONCLUSION

In this chapter, we have shown how to train a neural context retriever tailored for NER on a synthetic context retrieval dataset. Our retrieval system can be used to enhance the performance of a NER model, achieving an average gain of around 1 F1 point over a raw NER model and beating unsupervised retrieval heuristics. This performance gain heavily depends on the studied novel, as some novels benefit way more from context retrieval than others. We also demonstrated results on par or better than re-rankers trained using manually annotated data.

A limitation of our neural context retriever is that the relevance of each candidate context sentence is estimated separately. This is problematic, since it can lead to the model retrieving redundant context sentences instead of a set of sentences where each is useful. A possibility of improvement could be to iteratively add each candidate to the input sentence and rank the remaining candidates with respect to that newly formed input. Future works may expand on our method by exploring whether taking the candidate context sentences' relationships into account can have a positive impact on performance.

Since NER performance has a crucial impact on character network extraction, as we highlight in Chapter 3, our results are a step towards extracting better networks from novels. We also note that our proposed method, generating a retrieval dataset using an instructions-tuned LLM, could be leveraged for other NLP tasks where global document-level retrieval is useful. The only prerequisite needed to generate such a dataset is the knowledge of which types of context samples can help to improve the performance of a local task, in order to generate positive samples. Applying our method to other NLP tasks needed to extract character networks, such as coreference resolution, could therefore further improve extraction.

6 CHARACTER NETWORK APPLICATIONS

Contents

6.1 Analysis of Theater Plays: A Case Study on Lorenzaccio **65**
- 6.1.1 Character Network Extraction 66
- 6.1.2 Extracting Subplots with Community Detection 68
- 6.1.3 Mention and Conversational Networks 74
- 6.1.4 Takeaways 76

6.2 Narrative Alignment: A Case Study on A Song of Ice and Fire 77
- 6.2.1 Corpus 78
- 6.2.2 General Framework 79
- 6.2.3 Text-Based Method 81
- 6.2.4 Structure-Based Method 82
- 6.2.5 Hybrid Method 85
- 6.2.6 Takeaways 87

6.3 Conclusion **89**

In Chapter 4 and Chapter 5, we focus on improving the performance of NER to extract better character networks. In this chapter, we present contributions to network applications through two case studies:

- Literary analysis on Alfred de Musset's *Lorenzaccio* in Section 6.1. This section is based on our oral communication at the *Journées d'Informatique Théâtrales* [8].
- Narrative alignment on George R. R. Martin *A Song of Ice and Fire* and its adaptations in Section 6.2. This section is based on our journal article published in Social Network Analysis and Mining [9].

On both of these applications, we make use of dynamic networks, highlighting their usefulness despite their low usage in the character networks literature compared to static networks.

6.1 ANALYSIS OF THEATER PLAYS: A CASE STUDY ON LORENZACCIO

As we discuss in Chapter 2, character networks are largely used for literary analysis [110]. In this section, we focus on Alfred de Musset's *Lorenzaccio*, a French romantic theater play of 1834. Because of its complexity, *Lorenzaccio* was intended to be read rather than played, although it has seen live performances after the death of its author. The play adapts real events that occurred in the city of Florence in the 16th century. Florence, in this setting, is governed by Duke Alexander de Medicis, to the general resentment of the population. The story tells us about the efforts of different factions to try to overthrow or influence him. Lorenzo de Medicis, the main character, becomes close to Alexander and gains his trust for the sole purpose of murdering him to liberate the city. The Strozzi family, led by Philippe Strozzi, are fervent republicans, and hesitate throughout the play on whether to overthrow

Alexander. Meanwhile, Marquise Ricciardia Cibo, manipulated by Cardinal Cibo (her brother-in-law), schemes to become Alexander's lover and influence him. But, eventually, all of these characters' efforts are in vain: in the last act, despite Lorenzo killing Alexander, Cosmo de Medicis is appointed in his place by the Council of Eight, executing the order of the pope. The situation at the end of the play is almost the same as at the start: Florence is simply ruled by another Medicis that will apply the same politics.

When reading the play, it seems to be composed of disconnected subplots that correspond to the different conspiracies against Duke Alexander. Therefore, in this case study, we are interested in the following research questions:

- Can we identify these subplots?
- What is the relative importance of these subplots?
- Are there interactions between subplots?
- Can we automatically detect conspiracies against a character?

To answer these questions, we extract three different networks by extending Renard [20], our modular extraction library we presented in Section 3.3. Each of these networks models a specific type of interaction: co-occurrence, mention and conversation. We also leverage dynamic networks, which are not as often used as static ones in the literature. We work on the original French text of the play, but in this section, we translate the names of the characters in English for clarity. To do so, we base ourselves on the public domain translation of Thomson and Lorenzetta[1]. For reproducibility, we make our source code and data available under a free license[2].

6.1.1 *Character Network Extraction*

To extract character networks from *Lorenzaccio*, we use the version of the play available in TEI format through the DraCor project [73]. We develop several custom modules for our extraction library Renard, which allow us to process this TEI file. We manually correct a frequent issue in this file, where the same characters can have different identifiers. Moreover, we choose to group minor individual characters into a single collective character to limit their influence on the extracted networks. These are characters designated by a generic name (*"The Soldier"*, *"The First Cavalier"*) who appear in a few scenes and belong to a group containing several such characters. For instance, we group the various exiles from Florence (*"First Exile"*, *"Second Exile"*, *"Another Exile"*, etc.) into a single common vertex, *"The Exiles"*. Table 12 shows all these grouping operations. We also remove from the network the vertex *"the Eight"*, as it's a group character that appears only once and represents other characters that already exist in the network.

We extract three different static networks:

- A co-occurrence network (Figure 22a): This undirected network describes the joint participation of characters in the same scene. We connect two characters when they appear together in at least one common scene and weight this edge based on the number of scenes they share throughout the play.

1 https://labare.net/Lorenzaccio-EN/
2 https://github.com/CompNet/Lorenzaccio

Original Characters	Grouped Character
A Voice, Another Voice, First Exile, Second Exile, Third Exile, Fourth Exile, All The Exiles	The Exiles
The Guests, A Guest, Another Guest	Strozzi's Guests
The Citizen, First Citizen, Second Citizen, A Citizen, Second Citizen	The Citizens
First Schoolboy, Second Schoolboy	Two Schoolboys
First Nobleman, Second Nobleman	Two Noblemen
First Cavalier, Another Cavalier	Two Cavaliers
First Lady, Second Lady	Two Ladies
The Soldier, The Soldiers	The Soldiers
First Preceptor, Second Preceptor	Two Preceptors
A Voice in the Crowd, The Crowd	The Crowd
The Courtier, The Courtiers, Several Courtiers	The Courtiers

Table 12.: List of characters grouped into a single grouped character in our extracted character networks.

- A mention network (Figure 22b): This directed network represents the verbal mentions of characters. We add an edge from character A, who is speaking, to character B, whose name is explicitly mentioned by A. We weight this edge by the total number of such mentions across the play. While the speakers are explicitly annotated in the input file, mentioned characters are not. Therefore, to identify them, we use the French version of Renard's NER module that we present in Section 3.3.3. It relies on a CamemBERT model [133] that we fine-tune on the dataset presented by Alrahabi et al. [6]. Interestingly, in this network, only 3 characters talk about themselves: the main character Lorenzo, Philippe Strozzi and Marquise Ricciardia Cibo. This fact reveals their nature as romantic protagonists: the goal of the author is more to dress their psychological portrait rather than to depict their acts in relation to progressing the plot.
- A conversational network (Figure 22c): This directed network depicts direct verbal interactions between characters. Automatically determining which speaker addresses which listener is a difficult task, so we use a simple heuristic to approximate these interactions. We assume that when a character A speaks, they are addressing all other characters present in the current scene. We direct each edge from the speaker to the listener and weight it by the number of conversational interactions involving the same speaker and listener throughout the play.

For each type of interaction, we also extract dynamic networks at the level of scenes and acts. While it is possible to use different intervals, such as a certain number of consecutive scenes, we prefer to rely on a division that reflects an explicit choice by the author.

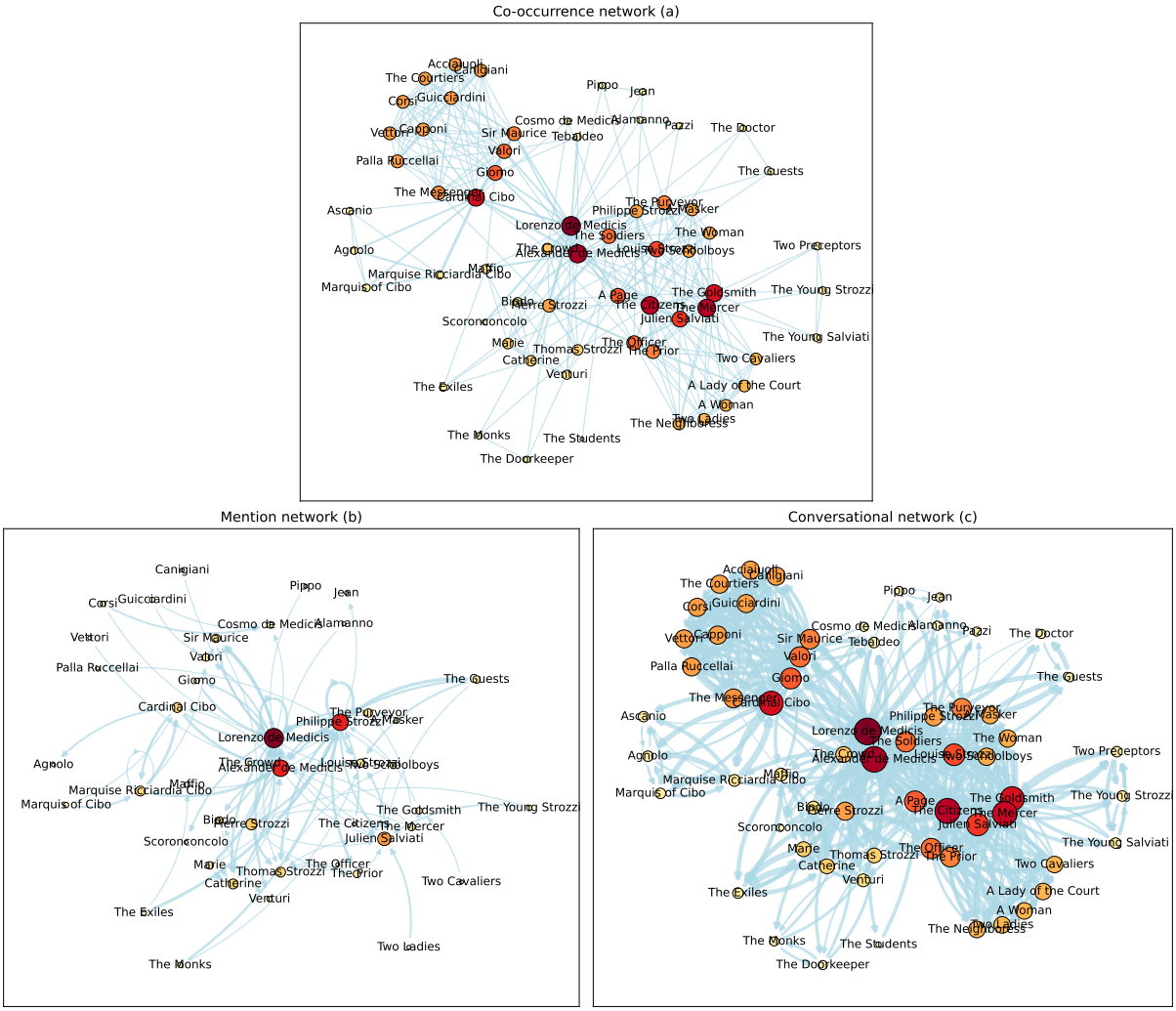


Figure 22.: Different static networks extracted from *Lorenzaccio*. The size of vertices and their color indicate degree. The thickness of edges indicates a) the number of scene co-occurrences b) the number of mentions c) the number of lines of dialogue between two characters.

6.1.2 *Extracting Subplots with Community Detection*

In this section, we aim to determine whether our networks reflect the division of *Lorenzaccio* into different subplots. Our initial hypothesis, based on our knowledge of the play, is that it is divided into three subplots:

1. Lorenzo's assassination of Duke Alexander,
2. Philippe Strozzi's internal conflict over using violence to free the city of Florence,
3. Marquise Ricciarda Cibo's attempt to influence Alexander's policies by becoming his lover.

Our goal here is to test this hypothesis by comparing it with the social structure depicted in our character networks, and to identify which subplots dominate the play.

To do so, we propose to observe the community structure of our static co-occurrence network, as it is already used in the literature to detect subplots involving groups of characters [41, 128, 153]. In our case, we extract communities of k-cliques from our co-occurrence network using the k-clique percolation method [150]. This technique has the advantage of

producing communities with possible overlaps: a vertex can therefore belong to multiple communities, which seems appropriate for stories where characters participate in multiple subplots, as is the case in *Lorenzaccio*. Some vertices may also belong to no community, which is fitting for certain minor characters weakly connected to the rest of the network.

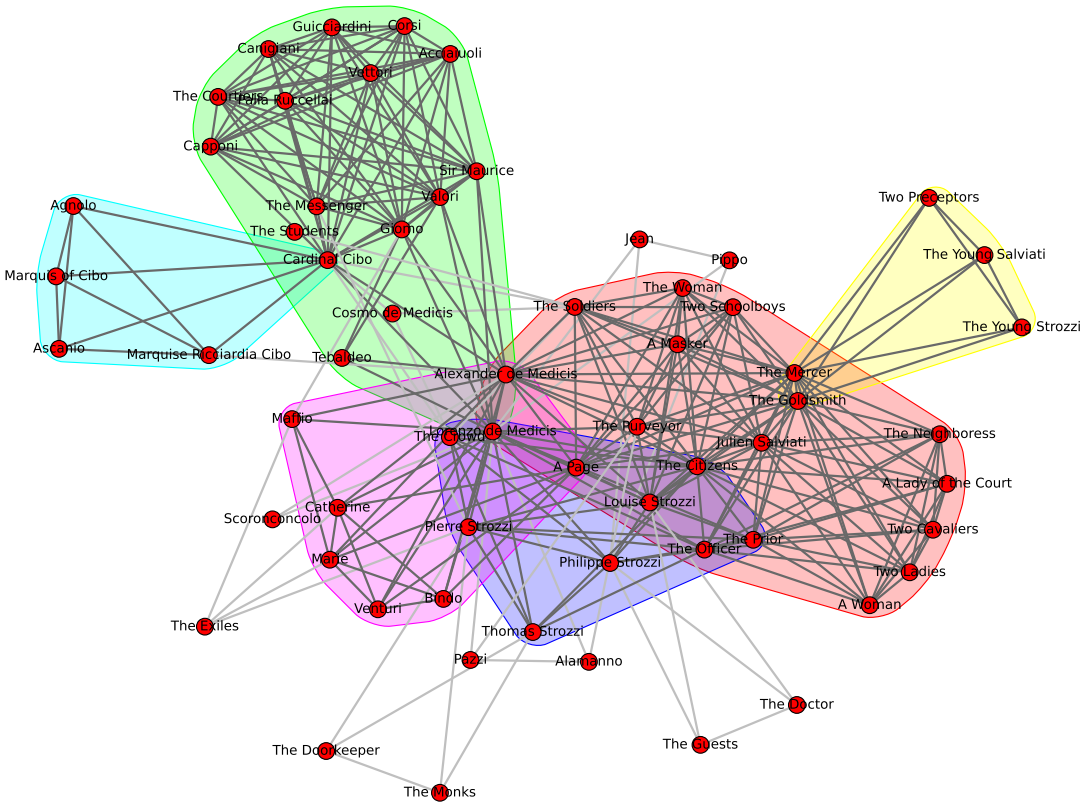


Figure 23.: *k*-cliques communities automatically detected in the static co-occurrence network of *Lorenzaccio* with $k = 5$. We keep the same layout as in Figure 22.

Figure 23 shows these communities for $k = 5$. We select this value of k because it is close to the average number of characters per scene (4.64), which seems reasonable for identifying characters interacting in at least one scene. Moreover, based on our knowledge of the play, our experiments with other values reveal that the communities extracted with a lower k are narratively too broad or too specific (false positives), while a higher k fails to capture some communities that we consider important (false negatives).

We interpret the communities as follows:

- **People of Florence:** The red community corresponds to the general populace of Florence, observing Duke Alexander's court in their daily lives. This is especially evident in the first act, where part of the exposition involves citizens watching and commenting on the lives of the nobles.
- **The assassination of Duke Alexander:** The purple community is linked to the subplot of the assassination of the Duke by Lorenzo. It includes these two characters, as well as others directly involved in the assassination, such as Catherine, whom Lorenzo uses to lure the duke to her room.

- **The Strozzi family:** The dark blue community corresponds to the Strozzi subplot, where they debate whether they should use violence to overthrow Alexander, under the leadership of their patriarch, Philippe.
- **Marquise Ricciarda Cibo** The light blue community is connected to the subplot of Marquise of Cibo. She, sometimes influenced by Cardinal Cibo, attempts to seduce the duke in order to influence his policies. The Marquise is weakly linked to the other communities, a possible sign that this subplot is peripheral.
- **The court and the Council of Eight:** The green community corresponds to the Council of Eight, the court, and the influence of the Holy Roman Empire over Florence. The Council of Eight is responsible, at the end of the play, for electing the new Duke of Florence under the Pope's influence, following the assassination of Alexander.
- **The Young Salviati and Strozzi:** Finally, the yellow community is peripheral and corresponds to a particular scene where two children from rival families (Young Salviati and Young Strozzi) fight in front of their tutors. The scene serves as a form of commentary on the city's tense atmosphere but has no direct influence on its events.

In summary, we observe two categories of communities. On the one hand, the red and yellow communities correspond to groups of characters who serve as a backdrop. They allow the author to depict the situation in Florence through the eyes of its inhabitants and to establish a certain atmosphere. On the other hand, the purple, dark blue, and light blue communities correspond to our three subplots: the assassination of the Duke, the Strozzi family, and the Marquise's scheme (though more peripheral). The green community also seems to belong to this second category, even though it is more of an event than a subplot, as it represents the conclusion of the play with the election of a new Duke.

Vertex Roles and Interactions Between Communities

To understand the interactions between communities, we can use the concept of *community role* of vertices to see how they connect or are specific to communities. This concept was introduced by Guimerà and Amaral [85] to describe the function of metabolites in the metabolic networks of different organisms, but it can be directly applied to character networks. The method involves projecting vertices into a two-dimensional parametric space. These dimensions are based on the notion of the *community degree* $k_i(u)$ of a vertex, i.e., the number of neighbors that vertex u has within a community C_i. The first dimension of this space, denoted z, is based on the degree of a vertex within its own community, also referred to as its *within-module degree*. To express this quantity relative to the community, it is normalized by calculating its z-score:

$$z(u) = \frac{k_i(u) - \mu(k_i)}{\sigma(k_i)}, \tag{15}$$

where C_i is the community of vertex u and $k_i(u)$ its within-module degree, and where the terms $\mu(k_i)$ and $\sigma(k_i)$ denote, respectively, the mean and standard deviation of the within-module degree for all vertices within the community C_i. The z measure quantifies how strongly the vertex u is connected to its own community. As a z-score, a negative value (respectively, positive) indicates a within-module degree below (respectively, above) the community average. The second dimension of the space is the *participation coefficient* P:

$$P(u) = 1 - \sum_{1 \leq i \leq K} \left(\frac{k_i(u)}{k(u)} \right)^2, \tag{16}$$

where K is the number of communities in the network, and $k(u)$ is the (total) degree of vertex u. The P measure quantifies how uniformly u is connected to the different communities: a value close to 0 indicates a vertex primarily connected to a single community, while a value close to 1 indicates a vertex evenly connected to all communities. As noted by [158], the maximum value of P depends on the number of communities considered.

In the case that Guimerà and Amaral [85] originally address, the considered communities are *disjoint*, but here we are dealing with *overlapping* communities. Since a vertex can belong to multiple communities at once, we have two options. The first is to artificially reduce the problem to disjoint communities by considering separately each community containing the vertex separately. The issue with this approach is that it associates as many values with the vertex as there are communities to which it belongs, making it difficult to interpret the results. The second option, which we adopt, is to adapt the z and P measures to the case of overlapping communities. For z, instead of considering that u belongs to a single community C_i, we substitute C_*, the union of all the communities containing u:

$$C_* = \bigcup_{u \in C_i} C_i. \tag{17}$$

We denote k_* as the community degree calculated over C_*, which allows us to generalize the notion of within-module degree. We can then reformulate z to obtain our modified version of the measure, denoted as z^*:

$$z^*(u) = \frac{k_*(u) - \mu(k_*)}{\sigma(k_*)}. \tag{18}$$

In the case of a vertex v that does not belong to any community, we define $z^*(v) = 0$. As for the participation coefficient P, directly applying the formula from Guimerà and Amaral [85] to overlapping communities may result in some neighbors being counted multiple times. This can lead to negative values, whereas the measure is normally defined within the range $[0; 1]$, which prevents proper interpretation of the results. To maintain the original bounds, we propose adjusting the normalization term by accounting for the multiplicity of neighbors:

$$k'(u) = \sum_{1 \leq j \leq K} k_i(u). \tag{19}$$

Next, we use $k'(u)$ as the denominator in the original formula to obtain our adjusted version, denoted as P^*:

$$P^*(u) = 1 - \sum_{1 \leq i \leq K} \left(\frac{k_i(u)}{k'(u)} \right)^2. \tag{20}$$

Our two measures z^* and P^* are equivalent to the original ones in the case where the communities are not overlapping.

Figure 24 shows the positions of the characters in the z^*–P^* space, using the same communities as in the previous section. While Guimerà and Amaral [85] empirically defines different vertex zones that correspond to different roles, we refrain from doing so as our measures behave differently. Instead, we proceed as Dugué et al. [64], and group vertices using the k-means algorithm [121], finding k with the elbow method. Colored zones in Figure 24 correspond to the 3 clusters we find. We make the following observations:

- Characters in the top right (red zone) of our z^*–P^* space have both an important within-degree module z^* and participation coefficient P^*. Guimerà and Amaral [85] refers to

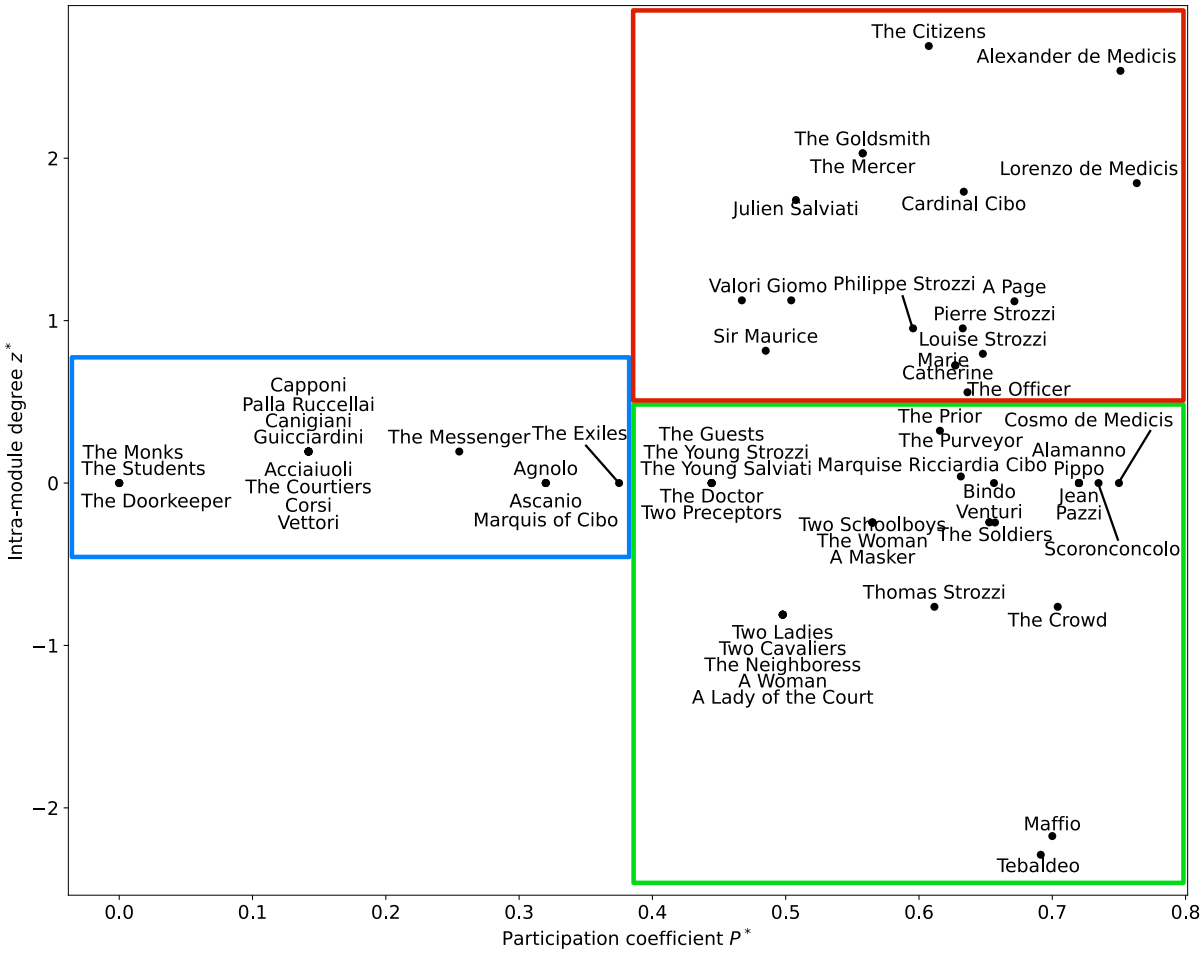


Figure 24.: Position of the characters in the z^*–P^* space. Several characters have the same position. Colored zones indicate groups of characters in space that we detect using the k-means algorithm [121] and the elbow method.

these types of vertices as *kinless hubs*. They connect different communities, and are therefore not clearly associated with a single module. We find two of the main characters in this group, Lorenzo and Alexander. Both appear in multiple communities (4 and 3, respectively) and can be said to connect different subplots. Overall, they have the highest participation coefficient of all vertices in the z^*–P^* space. In general, main characters of the story fall in this red cluster, except for a few ones like Marquise Ricciardia Cibo. Interestingly, some secondary characters are still present in the cluster, such as the Citizens, The Goldsmith and the Mercer. While they do not move the plot, they often appear to comment on Florence's political situation.

- Characters in the center-left of the space (blue zone) roughly correspond to what Guimerà and Amaral [85] call *peripheral vertices*. It mostly consists of minor characters, that appear in few communities, do not or weakly connect them. Interestingly, the Council of Eight is part of these vertices, isolated from the other communities, with its members having a low participation coefficient P^*. This isolation mirrors how the new Duke of Florence is chosen at the end of the play: the other characters have little influence on the decision, which is ultimately imposed by the Pope's will. Only a few minor characters (the Doorkeeper, the Students, and the Monks) have lower participation coefficients.

- Characters in the center right of the space (green zone) correspond to *non-hub kinless vertices*. While they are connected with different communities, they do not favor connections with one in particular, and are not strongly connected in their communities compared to other vertices. Many of these characters are minor, except for a few, like Marquise Ricciardia Cibo. for many minor characters, their large participation coefficient comes from the fact that they are connected to Alexander or Lorenzo, which are themselves in multiple communities.

Overall, we see that Alexander and Lorenzo are central characters: they are well-connected in their communities and connect multiple ones. Vertices with a low participation coefficient are specific to certain subplots, while other characters participate in different ones. Characters from the Council of Eight are clearly isolated, mirroring the low influence of other characters on their decision when appointing a new Duke. This isolation shows the lack of influence of the protagonists on the destiny of Florence. As is typical in romantic plays, the author spends time on their psychological aspects, but gives them really low agency over events.

Community Prevalence by Act

To understand the relative weight of the subplots and their distribution over time, we define the *prevalence* $p(i, j)$ of a community i within an act j:

$$p(i,j) = \frac{|C_i \cap A_j|}{|A_j|}, \tag{21}$$

where C_i and A_j are the sets of characters appearing in community i and act j, respectively. Prevalence represents the proportion of characters in the act that belong to the considered community. Since communities overlap, the sum of prevalences for all communities within the same act does not reach a fixed value, making p difficult to interpret. To compare communities within the same act, we define a normalized version $\bar{p}(j, i)$ of prevalence:

$$\bar{p}(j,i) = \frac{p(j,i)}{\sum_k p(k,i)}. \tag{22}$$

With $\bar{p}$, the sum of values for all communities equals 1 for any given act.

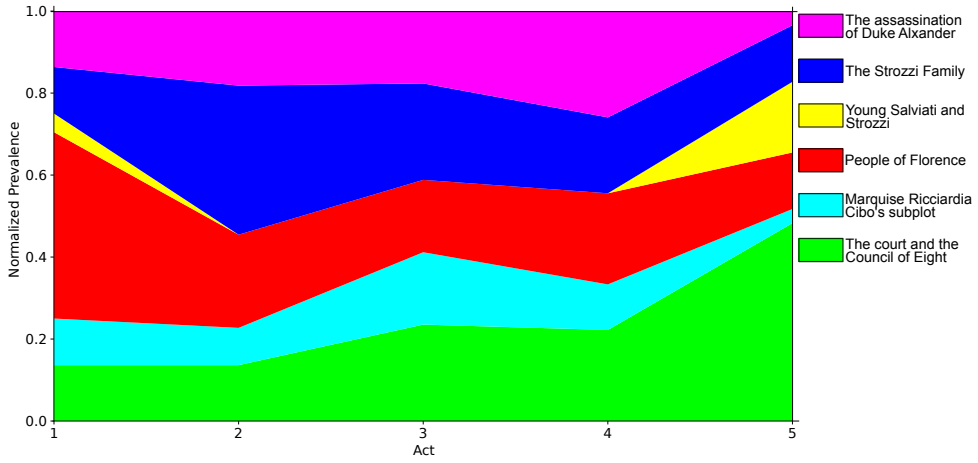


Figure 25.: Normalized prevalence of the different communities by act. Colors correspond to the community in Figure 23.

Figure 25 shows the normalized prevalence of communities across the five acts of the play. We observe the high prevalence of the community representing the people of Florence during

the first act, which serves to introduce the court. The Strozzi subplot dominates the second act, while the third act is more balanced. Act 4 marks the peak of the assassination plot, where Lorenzo kills the Duke. Finally, the court and the Council of Eight community dominate the last act, where the council decides the fate of Florence by choosing a new Duke. The subplot of the Marquise remains in the background throughout the play.

6.1.3 *Mention and Conversational Networks*

Underground Network

Figure 26 shows the network consisting of edges that appear only in the mentions network but not in the conversational network. We refer to these relationships as *underground*: the connected characters never speak directly to each other during the play but mention one another. In terms of degree, Philippe Strozzi is clearly the most central character in this network, with 8 neighbors. This confirms his role as an important figure of the agitation in Florence: characters sympathetic to his ideas are hoping he will make a move, and his enemies mention him as a potential menace. Meanwhile, Philippe mentions his enemies Julien Salviati and Duke Alexander. Julien Salviati is the character with the highest in-degree of the underground network. While this character does not appear very often in the play, he is a central point in Florence's tension: suspected of having killed Louise Strozzi, he is the origin of the Strozzi's revolt.

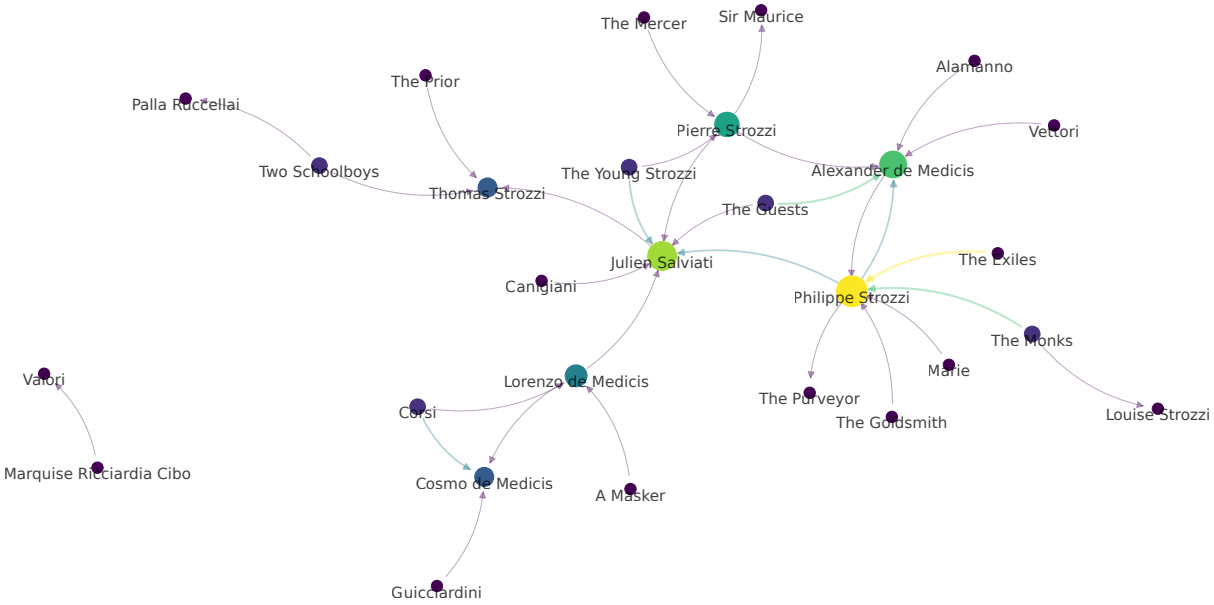


Figure 26.: Directed underground network formed by edges between characters that exist in the mention network but not in the conversational network.

Conspiracies

Lorenzaccio is a play where different conspiracies occur. We want to know if it is possible to automatically detect these conspiracies using networks. We define a conspiracy as a tuple (C, t), where C is an ensemble of at least two co-conspirators. We propose a simple detection algorithm using dynamic mention and conversational networks at the scene level.

Given a target character t, we create an initially empty conspiracy graph G_t. The algorithm runs through the networks of each scene where the target character is absent, and add a

vertex to G_t for each character that mentions t, and edges between all these characters. After running through all existing scenes, we consider connected components of G_t with a number of vertices greater than one as different co-conspirators groups. We repeat the process for all the possible target characters of the play. Using this algorithm, we extract the following conspiracies:

- ({*Alamanno, Philippe Strozzi, Strozzi's Guests, Lorenzo de Médicis*}, *Alexander de Médicis*): This conspiracy represents the scheme of Lorenzo and Philippe Strozzi, that plot the murder of Alexander. Alamanno is informed of the idea by Lorenzo during act IV, which includes him in the co-conspirators group.
- ({*Corsini, A Masker*}, *Lorenzo de Medicis*): Corsini is accidentally hurt by Lorenzo during the first act, as remarked by the masked character, and curses him. Ideally, this conspiracy should not be detected by our algorithm, as Lorenzo is physically present in the scene, but this is only indicated through the masked character's remark.
- ({*Philippe Strozzi, Strozzi's Guests*}, *Julien Salviati*): the Strozzi family effectively schemes against Julien Salviati, as they think he had a hand in the murder of Louise Strozzi.
- ({*Catherine, Marie*}, *Lorenzo de Medicis*): Catherine and Marie worry about the evolution of Lorenzo during the years.
- ({*Cardinal Cibo, Marquise Ricciarda Cibo*}, *Alexander de Medicis*): The algorithm correctly identifies Ricciardia Cibo's conspiracy.
- ({*The Exiled, Pierre Strozzi, Marie*}, *Philippe Strozzi*): The Exiled and Pierre criticize the choice of Philippe to not help them overthrow Alexander. Marie appears in this conspiracy because she is present in the same scene and mentions Philippe, even though she does not directly interact with the Exiled or Pierre.
- ({*Pierre Strozzi, The Prior*}, *Thomas Strozzi*): This conspiracy is also a false positive, due to the non-explicit presence of Thomas Strozzi in the scene.
- ({*Corsi, Giomo*}, *Lorenzo de Médicis*): Corsi and Giomo discuss, in the last act, of the betrayal of Lorenzo.
- ({*Corsi, Guicciardini*}, *Côme de Médicis*): In the last act, Corsi and Guicciardini mention Côme as a possible candidate for the throne of Florence.

A number of the extracted groups described above are not real conspiracies. Even if we except the false positives due to the approximation we use when extracting conversational networks, a group of characters can mention another one without scheming against him. Hence, we propose two ways to filter further our group of conspiracies: topic modeling and sentiment analysis.

TOPIC MODELING In order to keep only relevant conspiracies, we propose to filter the already detected conspiracies by checking the content of co-conspirators conversations with topic modeling. To do so, we use the BERTopic system [84]. It starts by representing each document as an embedding, and then applies a dimensionality reduction technique. Next, subjects are detected using clustering. Finally, a weighting scheme is used to represent each cluster with an ensemble of representative words. In practice, we treat each line of dialogue as a document. We use a multilingual SentenceBERT model [164] to represent these lines of dialogue, and UMAP [134] to reduce the dimensions of the resulting embeddings. We then

apply the HDBSCAN clustering algorithm [47], which has the advantage of filtering outliers. Finally, we use the KeyBERT-inspired weighting scheme included in BERTopic[3] to assign a set of keywords to each cluster. This scheme considers that a word is representative of a cluster when its embedding is close to the embedding of documents of the corresponding cluster. We avoid using stopwords as representative words by filtering them beforehand, using the French stopwords list of the NLTK [38] library. For each conspiracy, we extract the set of topics mentioned by co-conspirators when mentioning their target. These subjects allow us to describe conspiracies more precisely, and to filter them manually. For the conspiracy of Lorenzo and the Strozzi against Alexander, we obtain the following topics[4]:

- *tué, mort, tuerai, Alexander, croire* (killed, death, will kill, Alexander, believe).
- *Rome, Italie, traître, Lorenzo, despotisme* (Rome, Italy, traitor, Lorenzo, despotism).

Thus, this conspiracy is characterized by the semantic field of murder, betrayal and politics surrounding the city of Florence, making it easily detectable. Meanwhile, the conspiracy of Marquise Ricciardia Cibo and Cardinal Cibo against Alexander is composed of the following topics:

- *prince, roi, princes, seigneurs, tué* (prince, king, princes, lords, killed).
- *rêve, rêves, éveillé, autrefois, fantôme* (dream, dreams, awake, in the past, ghost).
- *évêque, empereur, prêtre, apostololique, roi* (bishop, emperor, priest, apostolic, king).

These topics do not evidently indicate a conspiracy, making it hard to detect, even manually.

SENTIMENT ANALYSIS While topic modeling is useful to manually filter conspiracies, it cannot be used to automatically do so. Therefore, we propose to use sentiment analysis. When constructing the conspiracy graph G_t for target character t, we add a condition to insert an edge between co-conspirators: their mention of character t must be negative. We check if this is the case using an adaptation of the VADER [98] sentiment analyzer for french[5]. Doing so, we obtain the following conspiracies:

- ({*Alamanno, Philippe Strozzi, Strozzi's Guests, Lorenzo de Medicis*}, *Alexander de Médicis*)
- ({*Corsini, A Masker*}, *Lorenzo de Medicis*)

The first one is the main conspiracy of the play, which is detected with success. The second one reveals the tension between Corsini and Lorenzo.

6.1.4 *Takeaways*

In this case study on *Lorenzaccio*, we were interested in 4 research questions:

CAN IDENTIFY SUBPLOTS? To answer this question, we used community detection to try to identify the different subplots of the play. We note two different kinds of communities: subplot-related ones and background ones.

3 https://maartengr.github.io/BERTopic/api/representation/keybert.html
4 We keep the five words deemed as most representative for each topic
5 https://pypi.org/project/vaderSentiment-fr/

WHAT IS THE RELATIVE IMPORTANCE OF SUBPLOTS? Using the community we extracted, we were able to compute the *prevalence* of a community in a given scene. This allowed us to conclude that, while several subplots are taking place at the same time, none of these really dominates the play. The subplot of the Marquise, however, clearly stays in the background.

ARE THERE INTERACTIONS BETWEEN SUBPLOTS? With the concept of *community role* [85], we observe that many vertices interact with different communities. The most central characters, Lorenzo and Duke Alexander, are themselves part of different subplots. Interestingly, members of the Council of Eight have a low participation coefficient, highlighting how the final destiny of Florence is unrelated to the protagonists' efforts.

CAN WE AUTOMATICALLY DETECT CONSPIRACIES? We proposed an algorithm to automatically detect communities using our dynamic mention and conversational networks. While it is able to detect some conspiracies, it also produces false positives. We propose two different methods to filter these false positives: a manual one using topic modeling, and an automatic one using sentiment analysis. Future research might be interested in investigating our proposed algorithm in a more systematic benchmark.

6.2 NARRATIVE ALIGNMENT: A CASE STUDY ON A SONG OF ICE AND FIRE

The process of adaptation consists in transcribing a story from a form (theater, TV show, novel...) into another [97]. When adapting a story, it undergoes a set of changes, because a particular adaptation has its own goals and because it must cater to the rules and conventions of the target form. Studying this adaptation process can uncover the reasons for an adaptation (which can be multiple: economic, making an homage, social commentary...) and the implicit conventions of the adaptation form. The changes occurring during the adaptation process involve different aspects of the story, one of these being its plot. For example, when adapting a novel into a movie, content may have to be cut to respect the usually expected screen time of a film.

To study the adaptation process in terms of plot, it would be useful to automatically match *narrative units* (novel chapters, series episodes...) from one version of a story to another, to better understand how their arrangement of events differ: in the literature, this task is called *narrative alignment*. *Pial et al.* [155, 156] tackle it using textual features. In this chapter, we propose to study the potential of character networks to perform narrative alignment. We proceed on a case study on *A Song of Ice and Fire*, the well-known series of fantasy novels by George R. R. Martin, and two of its adaptations: the *Game of Thrones* TV show and the comics adapted from the first two novels. We make available all the source code and data needed to reproduce our experiments[6].

We organize this section as follows. We first present our study corpus in Section 6.2.1. We then provide the general framework for our problem and approach in Section 6.2.2, before proposing and testing two narrative matching methods: one based on text (Section 6.2.3), and the other on character network structure (Section 6.2.4). Finally, we combine both approaches in Section 6.2.5, in order to assess their complementarity. Section 6.2.6 summarizes our findings.

6 https://github.com/CompNet/Sachan

6.2.1 *Corpus*

We explore narrative alignment on a corpus that encompasses three different media:

- **Novels:** The original *A Song of Ice and Fire* novels from George R. R. Martin. It consists of the 5 novels published at the time of this writing, but has yet to be fully completed. Novels are divided into chapters, each of them following the point of view of a single character.
- **TVShow:** The TV show *A Game of Thrones*, adapted by HBO from the original novels. Since the release of the TV show was faster than Martin's publication rhythm, it eventually caught on to the existing story in season 5. Season 6 starts to diverge from the original plot, while seasons 7 and 8 are mainly original compared to Martin's work.
- **Comics:** The comics (also titled *A Game of Thrones*) adapted directly from the first two novels. They consist of two volumes of 24 and 32 issues, respectively. Each issue is cut into chapters that each exactly adapt a chapter from the novels. These chapters are not ordered exactly as in the novels, probably in order to target a specific number of pages per issue.

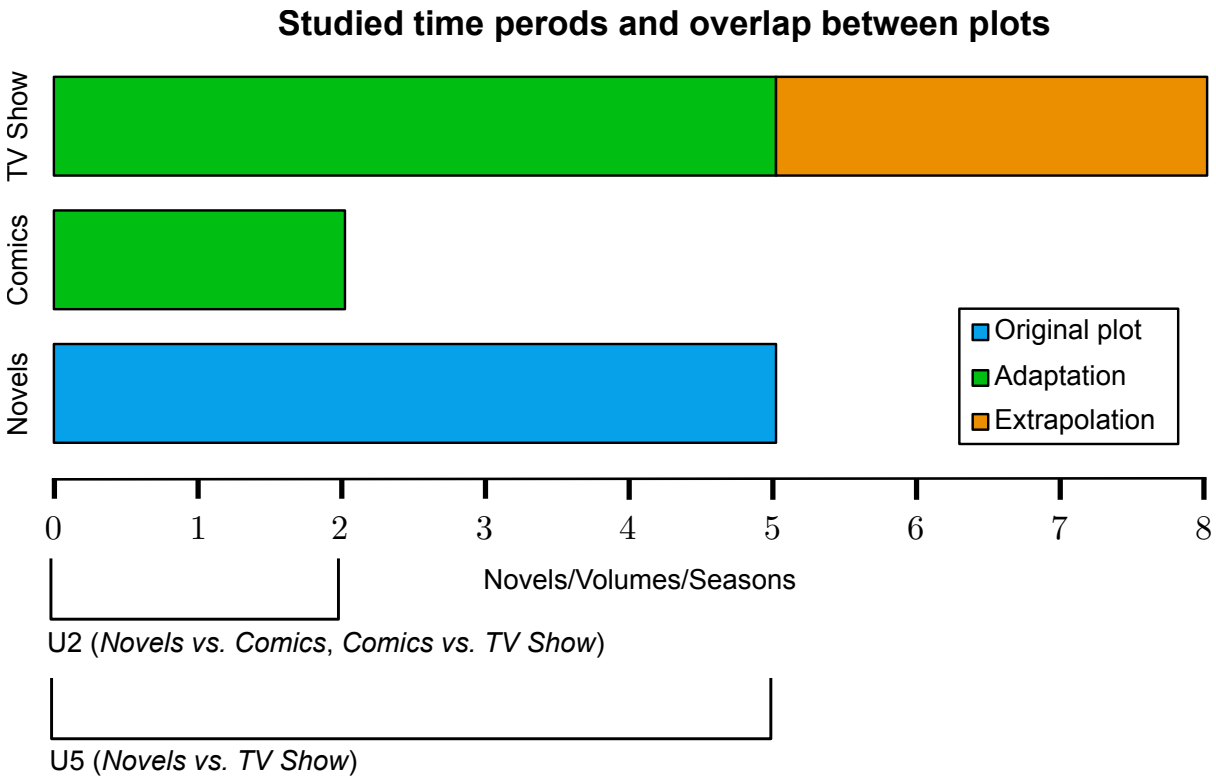


Figure 27.: Plot overlap between the three considered adaptations and studied periods U2 and U5.

For simplicity, we refer to these three works as *adaptations*. We perform narrative alignment on all the possible pairs of these adaptations: *Novels vs. Comics*, *Novels vs. TV Show* and *Comics vs. TV Show*. Since these media do not cover the same portion of events, we study two different periods: the first two novels, comics volumes, and TV show seasons, noted U2; and the first five novels and seasons, noted U5. We work on the largest common period for each pair. Since the TV show diverges more and more from the novels as time progresses, we additionally study the alignment of the *Novels vs TV Show* pair over the U2 period in the

Appendix (see Section A.2.3). Figure 27 shows the plot overlap for all three adaptations and time periods we consider.

Narrative unit	Novels	Comics	TV Show
Scenes	–	1,437	4,165
Blocks	–	–	739
Chapters	344	143	–
Episodes	–	–	73
Issues	–	56	–
Books/Volumes/Seasons	5	2	8

Table 13.: Numbers of narrative units for all three adaptations.

To perform narrative alignment, we need to define *narrative units*: each unit represents a self-contained part of the plot (such as a novel chapter or a TV show episode) that can be matched to another unit from another medium. We consider different narrative units depending on the media and the data we have available. For novels, we only consider chapters, are these are the only units we possess. For comics, we also consider chapters, since they correspond exactly to novels chapters. In some cases, we additionally consider issues, as we lack data to always perform matching at the chapter level. Finally, for the TV Show, we consider episodes and "blocks" when possible. A block is a custom narrative unit we define to try and better match novel chapters. Since episodes include content from several chapters, this renders matching more difficult because of the difference in scale. We define blocks as series of contiguous scenes from the TV Show that ideally correspond to a *point of view* sequence, i.e. a part of the story narrated from the perspective of a given character. We make the assumption that a new block in the TV show starts when there is a change in geographical location, as it is likely that this indicates that the focus switches to a different character group. We use the scene-level locations annotations from Jeffrey Lancaster [7] to automatically extract blocks by considering that two contiguous scenes are from the same block if they occur in the same location. Table 13 summarizes all the existing narrative units for all the adaptations.

6.2.2 *General Framework*

Problem Formulation

We define the narrative alignment problem as follows: given two media, let $\mathbf{a} = (a_1, ..., a_n)$ and $\mathbf{b} = (b_1, ..., b_m)$ be the vectors of their constitutive narrative units, with respective lengths n and m. For example, $\mathbf{a}$ could represent episodes from the TV show while $\mathbf{b}$ could represent chapters from the novels. We must find a matrix $\mathbf{M} \in \{0,1\}^{n \times m}$ where $M_{ij} = 1$ if narrative unit a_i corresponds to narrative unit b_i, and $M_{ij} = 0$ otherwise. $\mathbf{M}$ describes a many-to-many relationship: a narrative unit from $\mathbf{a}$ may correspond to several narrative units from $\mathbf{b}$, and vice versa. As an example, an episode from the TV show usually adapts several chapters from the novels, while parts of the same chapter may appear in distinct episodes. It is worth stressing that many-to-many matching is much harder than one-to-one matching, as the relation arity is unknown (i.e. one does not know how many narrative units are involved on either side).

7 https://github.com/jeffreylancaster/game-of-thrones

In order to evaluate our narrative matching methods, we gather gold alignments for all the pairs of media. After manual verification, we adapt the *Novels vs. TV Show* alignment from a fan annotation[8] who matched chapters and episodes. We create the *Novels vs. Comics* ourselves, at the levels of chapters. We use both of these to automatically produce the *Comics vs. TV Show* alignment. It is worth stressing that manually extracting such alignments necessitates significant effort, so proposing a method to automate this task with satisfying performance would be very useful.

When assessing a matching $\mathbf{M}$ given a gold standard matrix $\mathbf{M}^* \in \{0,1\}^{n \times m}$, we use F1-score, the harmonic mean of Precision and Recall. We define the following:

- A *True Positive* (TP) occurs when $M_{ij} = 1$ and $M^*_{ij} = 1$.
- A *False Positive* (FP) occurs when $M_{ij} = 1$ and $M^*_{ij} = 0$.
- A *True Negative* (TN) occurs when $M_{ij} = 0$ and $M^*_{ij} = 0$.
- A *False Negative* (FN) occurs when $M_{ij} = 0$ and $M^*_{ij} = 1$.

We refrain from using accuracy as the gold matrix $\mathbf{M}^*$ is sparse, which means that predicting 0 for all pairs of narrative units would result in a high accuracy despite the resulting matching $\mathbf{M}$ being useless.

Proposed Methods

Our general approach to perform automatic narrative alignment requires first to compute a similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times m}$ between the two considered media. We elaborate further on this point in the next subsections, but in summary, we experiment with two different types of information to obtain this matrix. First, we leverage textual representations of the three adaptations to derive a *textual similarity* matrix. Second, we take advantage of character networks to compute a *structural similarity* matrix. Finally, we also try to combine textual and structural similarity to compute a *hybrid similarity*, in order to understand if they are complementary.

After computing the similarity matrix $\mathbf{S}$ between two media, we use an alignment algorithm to derive a matching from it. We experiment with two different approaches: basic thresholding vs. the Smith–Waterman algorithm [184]. The former is straightforward: to produce a matching $\mathbf{M}$, we set a threshold t and we compute M_{ij} as follows:

$$M_{ij} = \begin{cases} 1, & \text{if } S_{ij} > t \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

For each pair of media, we estimate the optimal threshold t by tuning it on the two other pairs. As an example, to compute M_{ij} for the *Novels vs. Comics* pair, we use the value of t that gives the best F1-score for the *Novels vs. TV Show* and *Comics vs. TV Show* pairs. We consider thresholding as a simple alignment baseline.

The Smith–Waterman alignment algorithm [184] is a dynamic programming algorithm that was originally proposed to perform molecular sequence alignment. It identifies the best local alignment between two sequences by first scoring matches and gaps, and then backtracking through a dynamic programming matrix. Recently, Pial and Skiena [156] propose the GNAT alignment tool, and apply it to align plots between novels and the scripts of their

8 https://joeltronics.github.io/got-book-show/bookshow.html

film adaptations [155]. They modify the Smith–Waterman algorithm to allow for many-to-many matching. In their formulation, the Smith-Waterman dynamic programming matrix **H** is recursively computed as follows:

$$H(i,j) = max \begin{cases} H(i-1,j-1) + S_{ij}, \\ H(i-1,j) + g + max(0, S_{ij}), \\ H(i,j-1) + g + max(0, S_{ik}), \\ 0. \end{cases} \tag{24}$$

Where g is a gap penalty. To obtain an alignment from this matrix, we locate the highest-scoring cell and extend the segment left towards the start of the sequences until reaching a zero-score cell. In both articles, the authors limit themselves to textual similarity when computing the similarity matrix S. In order to also experiment with structural similarity, we re-implement their many-to-many adaptation of the Smith–Waterman algorithm. Similarly to the thresholding alignment method, we tune the parameters of the algorithm (such as the gap penalty g) for each pair of media on the two other media pairs.

In summary, our experimental setup involves three different ways of computing the similarity matrix between adaptations, and two algorithms to leverage this matrix and estimate the match, which makes a total of 6 methods. We apply them to the three possible pairs of media: *Novels vs Comics*, *Novels vs. TV Show* and *Comics vs. TV Show*. As stated in Section 6.2.1, each medium covers a different time period, so we work on the longest common period for each pair: `U2` for the *Novels vs. Comics* and *Comics vs. TV Show* pairs, and `U5` for the *Novels vs. TV Show* pair. For completeness, we additionally study the alignment of the *Novels vs TV Show* pair over the `U2` period in the Appendix (see Section A.2.3).

6.2.3 *Text-Based Method*

Our first method leverages textual representations of the adaptations to compute the similarity between them. We originally experimented with long textual representations: entire chapters, dialogues from TV Show episodes, and texts extracted using OCR for the comics. However, we found a much better matching performance when using the following short texts instead.

- **Novels Chapters:** We scrape the existing chapter summaries from the previously mentioned fan Wiki [1]. With a mean length of 4.56 sentences, these summaries are usually much shorter than episode summaries, and longer than comics summaries.
- **Comics Issues:** Most (41/56) issues of the comics contain a summary of the next issue that acts as a teaser. We use OCR to extract these texts, and correct them manually. For the 15 remaining issues, we manually write summaries of approximately the same length. The mean length of these summaries is 2.86 sentences, the shortest amongst all media.
- **TV Show Episodes:** We scrape the episode summaries of the first five seasons available from Wikipedia [217–221]. The mean number of sentences in these summaries is 11: since episodes encompass different points of views and a number of subplots, each of these is represented by one or more sentences.

These textual sources constrain the narrative units used to represent the plots: we perform alignment at the *chapter* level for the novels, at the *episode* level for the TV show, and at the *issue* level for the comics.

To compute the similarity matrix **S** between the narrative units of the pairs of media, we experiment with two different textual similarity functions:

- *tfidf*: compute the cosine similarity between the bag-of-words representations of two summaries weighted by standard TF-IDF [130]. This scheme weights each word according to its *term frequency* (TF) and *inverse document frequency* (IDF) [187], the latter being an indication of how characteristic of a document a term is.
- *sbert*: embed both summaries using SentenceBERT [164], and compute their cosine similarity. SentenceBERT is a variation of the language model BERT [60] specifically fine-tuned to extract semantic sentence embeddings, so that two sentences that are semantically similar should be close in the embedding space.

The results of our textual alignment can be found in Table 14. Matching performance varies highly between media pairs, with the *Novels vs. TV Show* pair being the hardest to match (best F1-score: 22.55) and the *Novels vs. Comics* being the easiest (best F1-score: 55.24). The Smith–Waterman alignment algorithm largely outperforms the thresholding baseline in almost all configurations. The role of the similarity function is more puzzling: while *tfidf* performs better when using the thresholding baseline, *sbert* outperforms it when using the Smith–Waterman alignment algorithm and obtains the best results. It is important to stress, however, that embedding summaries using SentenceBERT is computationally much more expensive than using TF-IDF.

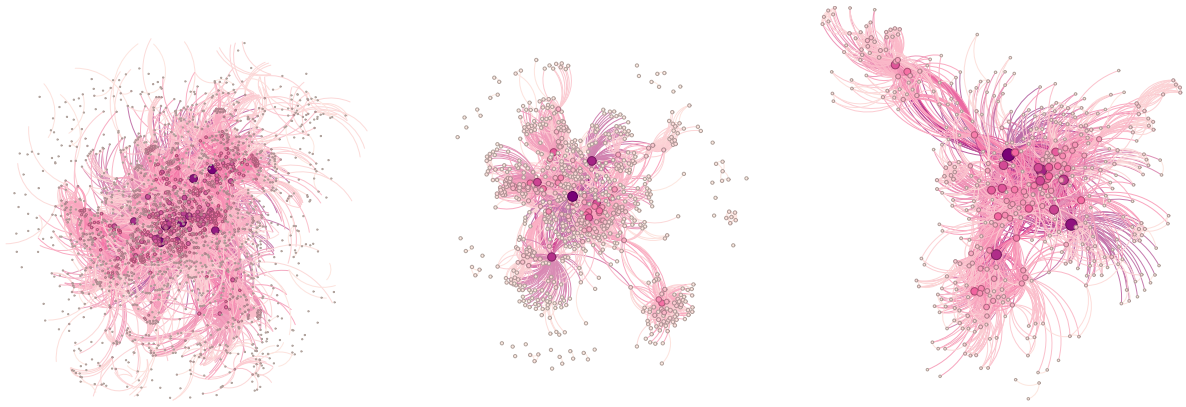
Table 14.: Performance obtained when using *text*-based representations to tackle the narrative matching task, expressed in terms of F1-score. Values in bold indicate the best performance for a pair of media.

Sim.	Alignment	*Novels vs. Comics*	*Novels vs. TV Show*	*Comics vs. TV Show*
tfidf	Thresholding	19.88	15.98	24.13
	Smith–Waterman	48.25	17.21	30.68
sbert	Thresholding	18.01	8.04	20.28
	Smith–Waterman	**55.24**	**22.55**	**35.96**

6.2.4 *Structure-Based Method*

Instead of using text as in the previous section, we now leverage character interactions to compute a similarity matrix between the narrative units of a media pair. To do so, we extract a dynamic network for each media:

- **Novels**: We use the manual annotations from Gessey-Jones et al. [79]. They applied a close-reading approach, and recorded each character appearing in a chapter, and every interaction between two characters in each chapter. Each edge in our network model interactions. This does include memories, as frequently important plot information is revealed through a character thinking of past events. As a result, the characters appearing in a chapter are not necessarily present at that particular time point.
- **Comics**: We manually annotate the comics at the scene level, using the method from Labatut [109]. Here, a scene is defined as a sequence of consecutive panels involving



(a) Novels (b) Comics (c) TV Show

Figure 28.: Static networks extracted from the three adaptations of *A Song of Ice and Fire*. The novels network is the biggest, with 1943 vertices overall.

the exact same group of characters, without interruption. In our network, we connect all characters interacting in the same scene.

- **TV Show**: Jeffrey Lancaster provides a very rich dataset concerning the TV show[9]. We leverage his annotations describing scene co-occurrences. As for the comics, we assume that two characters occurring in the same scene interact, and connect them in our network.

We choose the temporal window of these dynamic networks to correspond to the narrative units of interest. As we highlight in Section 2.1, there are two types of dynamic networks: instant vs. cumulative. The final cumulative networks are shown in Figure 28 for reference. We only present results obtained with instant networks, since they clearly outperformed cumulative networks in our experiments. We deal with the latter in the Appendix (see Section A.2.4).

Given a character network per narrative unit for two media, we experiment with several variants of Jaccard's index to assess their similarity. First, we consider the standard index computed over the sets of vertices present in the compared networks, as well as their sets of edges. Second, we use the *weighted* version of the index, a.k.a. Ružička's similarity [172]. The most straightforward approach to weight each vertex/edge is to use the number of occurrences of the characters/interactions. However, we obtain much better results when weighting according to the *inverse* number of occurrences. We present only these results in the following. We explain this difference by analogy with the IDF part of TF-IDF: using the inverse effectively gives more importance to less frequent characters/interactions. These are generally more typical of a narrative unit, compared to frequent characters, which are more likely to be involved in many narrative units. In the end, we have four ways of measuring the similarity, depending on whether we compare the vertex or edge sets, using the unweighted or weighted version of the index.

Computing all four of these similarity measures requires an exact mapping between characters from one media to another in order to obtain a meaningful score. To compute that mapping, we collect the names of each character among the different media and choose a single normalized name for each. Since some characters are not named, we also add a boolean attribute to all vertices to indicate whether they are named or not. Since we wish to understand whether minor characters are important to compute meaningful similarity between narrative units, we create three progressively more restricted subsets of characters: `named`,

9 https://github.com/jeffreylancaster/game-of-thrones

containing only the characters that are named, `common`, containing the named characters that are common to a pair of media, and `top-20`, restricting ourselves to the top-20 characters (ranked by the average proportion of their occurrences in their respective media).

In the rest of this section, we present the results of structural matching as follows. First, in the next subsection, we apply our methods to the same narrative units as with text in Section 6.2.3: novel chapters, TV show episodes, and comics issues. This allows us to compare directly between textual and structural matching. The structural approach is not bound to these specific units, though, as they were determined by the sources of the textual content in the first place. One possible issue with them is that they are not on the same scale, i.e. the chunks of plot they cover are not of comparable sizes. For example, TV show episodes usually adapt several chapters of the novels. To study the effect of the narrative unit scale, we also experiment with smaller, commensurate narrative units afterward.

Text-Constrained Narrative Units

The best results of structural alignment with the same narrative units as for text, using the thresholding and Smith–Waterman alignment methods, can be found in Table 15. We show the full results in Table 20 of the Appendix. The performance varies widely, with F1-scores ranging from 1.54 to 62.94, depending on the configuration. Overall, by choosing the best configuration, we obtain F1-scores of 62.94, 32.63 and 51.40 for the *Novels vs Comics*, *Novels vs. TV Show* and *Comics vs. TV Show* pairs, respectively. Structural matching leads to better performance than textual matching, for all three pairs of media: +7.7, +10.1 and +15.4 points, respectively. This is further confirmed in results for the *Novels vs. TV Show* pair over the `U2` period. As shown in Table 22 of the Appendix, structural matching obtains an F1-score of 45.00, while textual matching gets 36.65.

Table 15.: Performance obtained when using *structure*-based representations and the *text-constrained* narrative units to tackle the narrative matching task, expressed in terms of F1-score. Only the best results over all possible configurations are shown.

Alignment	*Novels* vs. *Comics*			*Novels* vs. *TV Show*			*Comics* vs. *TV Show*		
	`named`	`common`	`top-20`	`named`	`common`	`top-20`	`named`	`common`	`top-20`
Thresholding	29.61	34.74	13.31	20.72	23.43	7.34	43.36	46.95	33.33
Smith–Waterman	58.74	**62.94**	46.81	28.72	**32.63**	9.87	49.72	**51.40**	46.86

Based on these results, it appears that the Smith–Waterman algorithm largely outperforms thresholding for all configurations. Additionally, the `common` character set obtains the best results, while `top-20` severely underperforms and is almost always the worst option. It may be because more minor characters are specific to certain narrative arcs, while main characters are often together as groups, blending several narrative units together which renders matching more difficult. We cannot draw any conclusion regarding our Jaccard weighting scheme or whether computing Jaccard similarity on edges or vertices is better, as the behavior of these parameters is inconsistent across configurations (see Table 20 of the Appendix for more details).

Commensurate Narrative Units

The chapter constitutes the natural narrative unit of the original material (the novels), and the sole we have for this medium. Both other text-constrained narrative units, comic issues

and TV show episodes, have a larger scale. Put differently, the piece of plot they convey is longer than a chapter: both issues and episodes adapt several novel chapters (or, in the case of episodes, sometimes parts of chapters). We hypothesize this hinders the matching performance.

A solution is to adopt smaller narrative units that are closer to chapters. Scenes are available for both media, however, this unit is much smaller compared to novel chapters, which would cause the same scale difference problem as before. Instead, we turn to intermediary units: comic chapters, which are comparable to novel chapters, and TV show blocks, which are sequences of contiguous scenes happening in the same geographical location. We define the latter in an attempt to split the plot according to changes in character *point-of-views*, the literary device used in *A Song of Ice and Fire* novels to unveil the story.

Although we now compare the comics and TV show using chapters and blocks, we must come back to issues and episodes to conduct our assessment, in order to allow a fair comparison with the performance previously presented for the other matching methods. We do so by considering that a novel chapter matching a comic chapter or a block matches the whole issue or episode. Following the same principle, when matching *Comics vs. TV Show*, we match chapters and blocks, but consider issues and episodes to compute the performance.

Table 16.: Performance obtained when using *structure*-based representations and the *commensurate* narrative units to tackle the narrative matching task, expressed in terms of F1-score. Only the best results over all possible configurations are shown.

Alignment	*Novels* vs. *Comics*			*Novels* vs. *TV Show*			*Comics* vs. *TV Show*		
	named	common	top-20	named	common	top-20	named	common	top-20
Thresholding	25.66	33.23	14.04	21.02	18.72	7.74	36.25	36.71	29.51
Smith–Waterman	71.81	**72.30**	63.33	34.46	**35.09**	30.69	60.42	**61.78**	61.46

Table 16 shows the best alignment results using the commensurate narrative units. This scheme strongly increases matching performance when using the Smith–Waterman alignment algorithm, with gains of 9.4, 2.5 and 10.4 points for the *Novels vs. Comics*, *Novels vs. TV Show* and *Comics vs. TV Show* pairs, respectively. This performance gain is further confirmed over the U2 time period for the *Novels vs. TV Show* pair, with a large gain of 17.9 F1 (see Section A.2.3 in the Appendix. Interestingly, even though we extract TV show *blocks* automatically, which may induce errors, using these for matching still leads to better performance than with full episodes. Our results show that ensuring the *scale* of the narrative units used for alignment are comparable is important for performance.

6.2.5 *Hybrid Method*

In this section, we strive to combine our textual and structural methods, in order to assess their complementarity. For each pair of media, we adopt a direct approach that consists of computing a new *hybrid* similarity matrix $\mathbf{S}_h$, based on the structural and similarity matrices, respectively noted $\mathbf{S}_s$ and $\mathbf{S}_t$. We first rescale $\mathbf{S}_s$ and $\mathbf{S}_t$ separately using min-max normalization:

$$\mathbf{S}'_\star = \frac{\mathbf{S}_\star - \min(\mathbf{S}_\star)}{\max(\mathbf{S}_\star) - \min(\mathbf{S}_\star)}, \tag{25}$$

where $\mathbf{S}_\star$ denotes $\mathbf{S}_s$ or $\mathbf{S}_t$. We then combine the resulting matrices using a weighted sum:

$$\mathbf{S}_h = \alpha \mathbf{S}'_s + (1 - \alpha)\mathbf{S}'_t, \tag{26}$$

where α is a parameter controlling the relative importance of text vs. structure. As for our other parameters, we tune α for each media pair using the other two pairs as a development set.

While this combination method is pretty simple, we want to point out that we performed some additional exploratory experiments to combine textual and structural similarity, but that our attempts failed to improve over the results we present in this chapter. We experimented with training several machine-learning models to compute $\mathbf{S}_h$ from $\mathbf{S}_s$ and $\mathbf{S}_t$ instead of simply summing them. We also tried to perform early fusion by extracting embeddings from our dynamic character networks and combining them with SentenceBERT or TF-IDF vectors, and then experimented with multiple machine learning models to obtain S_c from the resulting representation.

As in the previous section on structural matching, we first experiment with matching using the text-constrained narrative units, and then using the commensurate narrative units (Section 6.2.5).

Text-Constrained Narrative Units

Table 17 shows the results obtained when applying our hybrid method on the text-constrained narrative units already used in Sections 6.2.3 and 6.2.4. As we experiment with all possible configurations of structural and textual matching, we only report the best performance for the sake of simplicity.

Table 17.: Performance obtained when using *hybrid* representations and the *text-constrained* narrative units to tackle the narrative matching task, expressed in terms of F1-score. Only the best results across configurations are shown.

Alignment	*Novels* vs. *Comics*	*Novels* vs. *TV Show*	*Comics* vs. *TV Show*
Thresholding	39.60	26.90	46.75
Smith–Waterman	**67.37**	**30.95**	**49.16**

We find that combining information from textual and structural matching increases performance for the *Novels vs. Comics* pair (+4.4 F1), but fails to improve performance for the *Novels vs. TV Show* (−1.7 F1) and *Comics vs. TV Show* (−2.2 F1) pairs.

Commensurate Narrative Units

As highlighted in Section 6.2.4, performing structural matching on narrative units of comparable sizes can strongly increase performance. Therefore, we now want to conduct hybrid matching by combining text with the commensurate units from Section 6.2.4. Our combination method requires summing the textual and structural similarity matrices $\mathbf{S}'_s$ and $\mathbf{S}'_t$, and therefore, we need the matrices to be of the same dimension. It is not the case though: as already explained, the text-constrained units have a larger scale, and the corresponding similarity matrices are consequently smaller. Moreover, it is not possible to build textual representations on a smaller scale: for the TV show, episodes summaries can not easily be cut to correspond to the underlying blocks, and comics issues summaries cannot easily be split to apply to underlying chapters. Therefore, we propose to artificially *extend* the textual

similarity matrix in order to match the dimension of its structural counterpart. We do so by duplicating the rows and columns corresponding to a text-constrained narrative unit (e.g. TV episode) as many times as the number of commensurate units it contains (e.g. blocks).

Table 18.: Performance obtained when using *hybrid* representations and the *commensurate* narrative units to tackle the narrative matching task, expressed in terms of F1-score. Only the best results across configurations are shown.

Alignment	*Novels* vs. *Comics*	*Novels* vs. *TV Show*	*Comics* vs. *TV Show*
Thresholding	47.62	23.58	46.24
Smith–Waterman	**78.50**	**39.04**	**63.87**

Table 18 shows the best results of hybrid matching on commensurate narrative units. As with purely structural matching, working with these narrative units greatly increases performance. Compared with hybrid matching on text-constrained units, we observe gains of +11.1, +8.1 and +11.3 F1 points on the *Novels vs. Comics*, *Novels vs. TV Show* and *Comics vs. TV Show* pairs respectively. Compared with purely structural matching using commensurate units, we also observe improvements across the board: +5.2 F1 point for the *Novels vs. Comics* pair, +4 points for the *Novels vs. TV Show* pair and +2.1 points for the *Comics vs. TV Show* pair. Overall, matching using hybrid similarity on the commensurate narrative units leads to the best performance.

6.2.6 *Takeaways*

The way we formalize the narrative matching task makes it difficult: we perform many-to-many matching and use F1 as a metric, meaning any mismatch is strictly penalized. Even so, our best results (78.50 and 63.87 F1 for the *Novels vs. Comics* and *Comics vs. TV Show* pairs, see Figure 29) show that our proposed alignment method can obtain good performance. Through our experiments, we are able to derive several important insights valuable to the application of the method. First, we note that structure-based matching using dynamic instant character networks outperforms text-based matching, highlighting the usefulness of networks for the task. To our knowledge, this is the first time that character networks are used for narrative alignment. We also show that combining text-based and structure-based similarities can yield better performance than using either alone. Furthermore, we demonstrate the importance of aligning stories using a comparable narrative scale, as taking commensurate narrative units into account consistently improves our results. Table 19 summarizes our best results.

Table 19.: Best configurations for the task of narrative matching using the *text-constrained* and *commensurate* narrative units.

Narr. Unit	Adaptations Pair	Similarity	Repr.	Measure	Character Set	Text Sim.	Alignment	F1
Text-Constr.	*Novels vs. Comics*	Hybrid	Vertices	Ružička	`common`	*tfidf*	Smith–Waterman	67.37
	Novels vs. TV Show	Structural	Edges	Jaccard	`common`	–	Smith–Waterman	32.63
	Comics vs. TV Show	Structural	Vertices	Ružička	`common`	–	Smith–Waterman	51.40
Commens.	*Novels vs. Comics*	Hybrid	Edges	Ružička	`named`	*tfidf*	Smith–Waterman	78.50
	Novels vs. TV Show	Hybrid	Vertices	Ružička	`common`	*tfidf*	Smith–Waterman	39.04
	Comics vs. TV Show	Hybrid	Edges	Jaccard	`top20`	*sbert*	Smith–Waterman	63.87

The results on the *Novels vs. TV Show* pair over the `U5` period are, however, more lackluster (39.04 F1 at best). We attribute this lower performance to the plot divergence between the

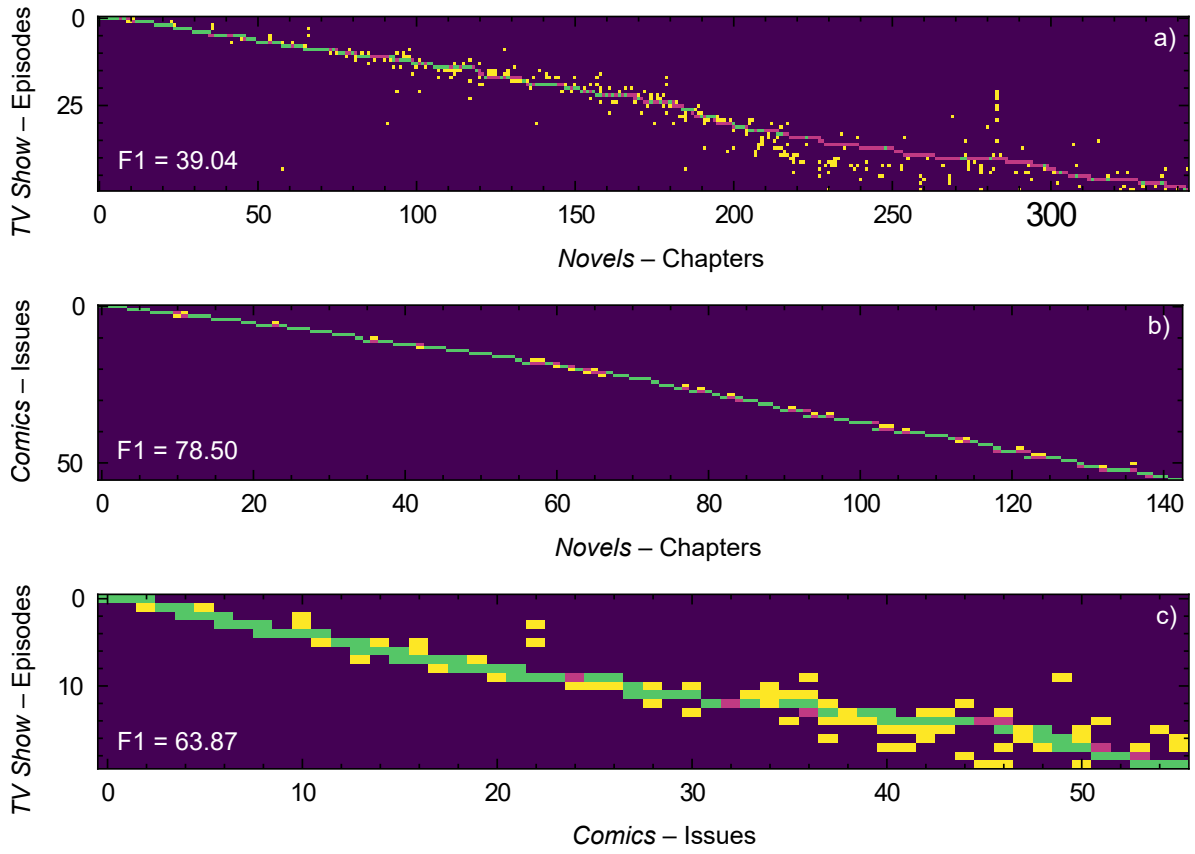


Figure 29.: Best performing alignment for all pairs of media, from top to bottom: a) *Novels* vs. *TV Show*, b) *Novels* vs. *Comics*, and c)*Novels* vs. *Comics*. Green denotes true positives, red false positives, yellow false negatives and purple true negatives.

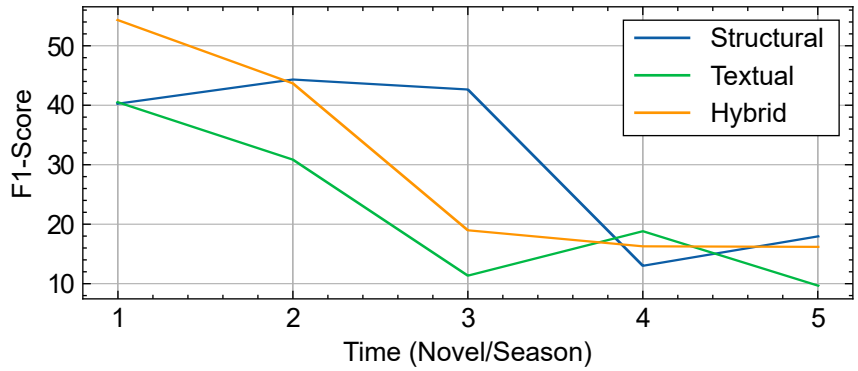


Figure 30.: Performance of the best configurations of narrative alignment over seasons for the *Novels vs. TV Show* pair, expressed in F1-score. As the TV show progressively diverges from the novels, the matching performance starts to decrease. F1-score for the last two seasons does not exceed 20.

original novels and their TV show adaptation, which increased over time. As a confirmation, results restricted to period U2 are much better, with a top F1-score of 64.65 (see Figure 31 and Table 22 in the Appendix). As shown in Figure 30, the matching performance progressively decreases as the TV show progresses for the structural, textual and hybrid similarities alike. Plot divergence is difficult to tackle for the Smith–Waterman algorithm, originally developed to align molecular sub-sequences: it assumes that parts of the sequences it tries to align are

ordered similarly, but this assumption is challenged on the *Novels vs. TV Show* pair. As seen in Figure 29, the blocks constituting the later seasons are ordered completely differently from their chapter counterparts, and some chapters are not even adapted, leading to low performance.

6.3 CONCLUSION

In this chapter, we proposed contributions to character networks applications on two case studies.

In our first case study on *Lorenzaccio*, we used three different networks to gain insights into the play: a co-occurrence network, a mention one and a conversational one. We used the co-occurrence network to detect subplots through community detection, and to comment the distribution of these subplots through time. We also used our mention and conversational networks to try to automatically detect conspiracies. While the algorithm we propose can detect conspiracies, it produces false positives, and filtering through them proved to be difficult.

In our second case study on *A Song of Ice and Fire*, we tackled the task of *narrative alignment* on three different media: the original novels, the HBO TV show and the comics. We reproduced the Smith-Waterman based matching algorithm of Pial et al. [155] and Pial and Skiena [156], and compared text-based or network-based alignment methods. We showed that network-based methods outperformed text-based one, but also that they could be combined to obtain better results. We also showed that performing alignment on *commensurate* units proved to increase performance, even when these commensurate units are extracted approximately.

In both cases, we made use of dynamic character networks: at the scene and act level for *Lorenzaccio*, and at the chapter, episode and issue level for *A Song of Ice and Fire*. Our two case studies demonstrate the usefulness of dynamic networks, that see a lack of usage in the literature.

7 CONCLUSION

Contents

7.1 Contributions 90
7.2 Perspectives 92

This chapter is organized as follows: we list our contributions in Section 7.1, and then turn to possible perspectives in Section 7.2.

7.1 CONTRIBUTIONS

In this thesis, we generally contributed to the field of NLP, and to character network extraction and analysis. We first proposed a step towards understanding the performance of character network extraction pipelines. We did so by developing two resources: a literary NER dataset, Novelties, and a character network extraction pipeline, Renard [20]. Our dataset is important in the current landscape of scarce existing literary resources. It also has the advantage of comporting several literary styles, when existing datasets are often comprised of classical novels in the public domain. As for our pipeline, we designed it based on existing practice in the character networks community [110], and made it modular to assess the impact of NLP tasks performance on the extracted networks. Using our Novelties dataset, we surveyed its performance and compared it with end-to-end extractors based on LLMs in a zero-shot setting. To do so, we proposed network quality measures: Vertex F1, Edge F1 and Weighted Edge F1. We observed decent performance of Renard on vertex extraction (70.16 Vertex F1), but lower performance on edge measures (43.66 Edge F1 and 32.41 Weighted Edge F1), which speaks for the difficulty of the task. We found Edge Recall in particular to be very challenging (36.10 Edge F1 and 26.53 Weighted Edge F1). In general, Renard outperformed LLMs extractors, which struggled even more on Recall. To go further, motivated by the lack of literature on the question of character network extraction performance, we also conducted an analysis to understand the impact of NLP tasks performance on the quality of extracted networks. To do so, we progressively degraded gold predictions for two NLP tasks, coreference resolution and NER, while observing the impact in terms of extraction measures. We found that while coreference resolution is important to network quality, some errors are more impactful than others. In particular, predicting wrong coreference links is very detrimental for character extraction performance, but missing a link mainly impacts edge measures. This suggests that a coreference model with high link precision might be sufficient to correctly extract characters, even if it has low recall. For NER, we found that its performance is crucial in general to extract qualitative networks, in terms of Precision as well as Recall. These findings will allow future research to focus their efforts on the most important aspects of extraction pipelines.

Our error analysis led us to propose enhancements for the NER task in the literary setting. We tackled two different problems: first, in Chapter 4, the case of cross-domain NER, a setup where a model is trained on a dataset from a domain and evaluated on one from another domain. This situation can arise when using a generalist literary NER model on an unseen literary genre. We identified that a potential problem in that setup is that some entity names of the unseen genre are entirely novel for the model, leading to detection issues. This confirmed

an earlier observation by Dekker et al. [59], who remarked that replacing these difficult to detect names by more common ones would drastically improve detection, showing that the name detection problem was caused by their form. To alleviate it, we proposed to use a data augmentation technique, mention detection, to inject unseen names of the target domain in the training dataset. We showed that this leads to improved NER Recall on the fantasy subset of the Novelties dataset. It results, however, in a decrease of Precision: we show that the cause of this decrease is the model being encouraged to predict ambiguous entities as `PER` due to our augmentation scheme, which can be alleviated by increasing the context seen by the model at inference. Second, we tackled the range limitations of transformers in Chapter 5, that occurs because of the computational costs of their attention mechanism. To reduce these costs, the commonly accepted practice is to process texts block by block in a rolling window fashion, which can lead to loss of performance [17]. One existing solution is to increase the size of transformers' context window by reducing the complexity of attention [193], but very long sequences can still be hard to process [192]. Therefore, we proposed to retrieve relevant context in the current document to assist the NER model when performing prediction. Since no retrieval dataset exist to train a model for that retrieval task, we suggested generating a synthetic one using the Alpaca LLM [191]. Through experiments on Novelties, we showed that using a neural retriever trained on this synthetic dataset at NER inference time generally increases NER performance. This improvement depends on the tested novel: some novels strongly benefit from retrieval, while others do not. We also compared the performance when generating the dataset with LLMs of different sizes (Alpaca-7b vs. Alpaca-13b), but we found that using more parameters did not improve performance. Qualitatively, we observed no difference in the generated datasets between Alpaca-7b and Alpaca-13b. Finally, we highlighted the impact of the size of the context window (i.e. the window in which a retriever can retrieve relevant context), which depends on the considered retriever. For our neural retriever, we found that retrieving context in the whole novel was more beneficial.

After focusing on extraction performance, we highlighted the interest of character networks by leveraging them on two different applications. First, we employed them to analyze Musset's *Lorenzaccio*, a 19th century Romantic French play. We performed community detection to unveil the different subplots of the play, to quantify their relative importance, and to find interactions between them. Motivated by the fact that some characters appear in multiple subplots, we did so using the k-cliques percolation technique [150], which has the advantage of extracting potentially overlapping communities. This led us to adapt the within-module degree and participation degree measures, originally proposed by Guimerà and Amaral [85] to analyze vertex roles in relation to modules, to the overlapping communities case. We also proposed a method to automatically detect conspiracies using the dynamic mention and conversational networks we extracted. Then, we applied character networks for the first time to narrative alignment, on a case study on three adaptations from George R. R. Martin's *A Song of Ice and Fire*: the original novels, the comics directly adapted from them and the popular HBO TV Show *Game of Thrones*. To tackle that task, we adapted the technique proposed by Pial et al. [155, 156] based on the Smith–Waterman alignment algorithm [184] to dynamic character networks. We showed that using character networks outperforms text-based techniques, and that combining both could bring performance improvements. Through our results, we also demonstrated that using comparable narrative units for alignment is important for performance. In both mentioned applications (literary analysis and narrative matching), we made use of dynamic character networks, demonstrating their usefulness in practice despite their low usage in the literature.

Our work in this thesis prompts for exploring some additional questions, and also suggests some research directions that we want to present in this section.

First, we remarked in Chapter 3 that NER and coreference resolution both have an important impact on the performance of character network extraction. We largely focused on improving NER performance in this thesis, but coreference resolution is still a challenging task that is far from being solved. Further research into character network extraction should focus on performance improvements for that task, but another issue is applying coreference systems to long texts. Models based on the end-to-end paradigm introduced by Lee et al. [113] are very resource-intensive on such texts, while incremental models [86, 196, 226] are more efficient but lag behind on evaluation measures. In any case, systems struggle with performance at scale [86, 87]. Developing systems towards being both performant and efficient on long texts is therefore an important objective. Recently, two research directions have caught our attention on that front. First, Guo et al. [86] propose an incremental model based on a local and a global cache. Their system merge mentions by reading the text left-to-right, and progressively merging them with an existing coreference chain from these caches, or creating a new chain. The local cache keeps recently active chains in memory, while the role of the global cache is to recall frequently occurring chains that are recently unseen. While their system is successful at improving performance at scale, their long document benchmark is limited to a single novel due to the lack of existing annotated data. Annotating a novel-scale coreference dataset would be the first step to carry a study on the effectiveness of their proposed dual cache, and perhaps to find another cache mechanism to further improve performance. Second, Gupta et al. [87] propose a hierarchical merging mechanism for long-range coreference, that we included to some extent in our coreference library Tibert. They apply the word-level coreference system of Dobrovolskii [63] (itself an improvement of the end-to-end system of Lee et al. [113]) using a rolling window, effectively extracting chains block of text by block of text. Then, they merge local chains together by computing their representation as the mean of their mentions' embeddings, and using their model to decide whether they should be merged. While their model is more efficient than the original word-level coreference system [63], their long document benchmark is limited to two long texts, and we again conclude that a larger dataset would be a first step into a better understanding of their method performance characteristics. We also note that they directly adapt an existing model when performing hierarchical clustering: it is possible that a specialized merging model would perform better.

As we've just seen for coreference resolution, there is a lack of datasets in the literary domain. This is a problem on two aspects: the performance of NLP tasks suffers directly from that scarcity, and evaluating character network extraction is difficult given the lack of a standardized benchmark. In the case of NLP tasks, while a few datasets exist (such as Litbank [29, 30, 183], PDNC [205] or our proposed Novelties [10, 11]), most do not contain full novels. This is particularly an issue for coreference resolution, where performance at scale is difficult to obtain, as mentioned above. In the case of character network extraction, we relied on a converted version of Litbank for our extraction performance experiments in Chapter 3, but that method has its drawbacks. First, since Renard performs flat NER but Litbank is concerned with nested NER, we had to automatically convert its annotations, leading to a loss of information and potential mistakes. Second, Litbank excerpts are shorts (approximately 2,000 tokens each), and results obtained on such a dataset may not be representative of results on entire novels. A problem caused by this shortness is the potentially low number of character mentions, which makes it very easy to not detect them when an error occurs in the pipeline.

In general, the lack of network extraction datasets is problematic, as it forbids comparing extraction systems. Therefore, we prompt for the future creation of literary datasets, particularly character network extraction dataset, preferably on entire novels.

We also observe that current network extraction pipelines are of the cascading type. This means, as we discussed in Chapter 3, that errors may propagate along the pipeline, which can lead to lower performance. A potential solution to this issue is to consider end-to-end pipelines. We surveyed some LLMs in a zero-shot end-to-end setting in this thesis, and while Renard performed better overall, they have shown promising performance. Further research may be interested in fine-tuning such models to improve performance or in exploring different prompting directions, for example, applying chain-of-thought [212] or tree-of-thought [122, 233] prompting. End-to-end text-to-graph models may also be of interest, but are not explored in the literature when it comes to extracting character networks. For example, recently, Zaratiana et al. [238] propose *GraphER*, a model tackling IE as graph structure learning.

Character networks are used in many applications: classification, summarization, recommendation systems... However, there are many graph-related aspects that are under-exploited for these tasks. This is the case of dynamic networks, which we leveraged in our case studies in Chapter 6. Signed networks and additional vertex or edge attributes are also rarely utilized. In general, there are many existing concepts and measures in the complex networks literature [55, 178] that have yet to be used when it comes to character networks, as authors generally focus on the most common ones. Finally, recent graph learning techniques are also relatively unexplored: for example, as we remarked in Chapter 2, only traditional graph learning techniques are employed in the literature on classifying networks extracted from novels. Comparatively, more recent techniques such as GNNs are not as studied but could bring performance improvement for different applications.

While we focused on text in this thesis, character networks can represent contents of different modalities. We made use of this multimodal representation property for narrative matching in Chapter 6, where we were able to align narrative units from different media using dynamic networks. We think this property is particularly interesting for some tasks: for example, a recommendation system might work across different media because their representation is unified. Existing multimodal systems such as multimodal LLMs necessitate an alignment phase between modalities [57], which would not be necessary in the case of character networks as they represent the same concepts across modalities.

Finally, and to expand further on that previous perspective, the fact that different types of contents can be represented by character networks means that many networks can be extracted and collected. In the light of the success of self-supervised models trained on massive amount of data (such as BERT [60] or, more recently, LLMs), we suggest that this data may support training such a model. Self-supervised learning on graphs already exists, using tasks such as structural or feature reconstruction [96, 116], and graph foundational models have recently been developed for certain domains [132]. A foundational model trained on a large scale character networks dataset may have a lot of applications for tasks that can take advantage of relationships structure between sets of entities. A potential question is the influence of the quality of the networks on such an endeavor, a question we leave to future research.

BIBLIOGRAPHY

[1] A Wiki of Ice and Fire. *A Game of Thrones (comics)*. Nov. 2021. URL: https://awoiaf.westeros.org/index.php/A_Game_of_Thrones_(comics).

[2] A. Agarwal, A. Corvalan, J. Jensen, and O. Rambow. "Social Network Analysis of Alice In Wonderland." In: *NAACL - Workshop on Computational Linguistics for Literature*. 2012. URL: http://www.cs.columbia.edu/~apoorv/Homepage/Publications_files/naacl2012.pdf.

[3] A. Agarwal, A. Kotalwar, J. Zheng, and O. Rambow. "Sinnet: Social interaction network extractor from text." In: *International Joint Conference on Natural Language Processing - System Demonstrations*. 2013, pp. 33–36. URL: http://www.aclweb.org/anthology/I13-2009.

[4] C. Aggarwal and K. Subbian. "Evolutionary Network Analysis: A Survey." In: *ACM Computing Surveys* 47.1 (2014). DOI: 10.1145/2601412.

[5] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf. "FLAIR: An easy-to-use framework for state-of-the-art NLP." In: *Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 2019, pp. 54–59.

[6] M. Alrahabi, C. Brando, F. Frontini, A. Provenier, R. Jalabert, M. Bordry, C. Koskas, and J. Gawley. "Guide d'annotation manuelle d'entités nommées dans des corpus littéraires." In: *HAL* (2021). URL: https://ens.hal.science/ODHN/hal-03156278v1.

[7] A. Amalvy. "Natural Language Processing applied to Interactive Character Relationships Visualization in Novels." MA thesis. National Central University, 2020. URL: https://cloud.klmp200.net/index.php/s/f7YRqcW9oAwwLKd.

[8] A. Amalvy, N. Diassinous, and V. Labatut. "Dynamique des relations dans Lorenzaccio : modélisation par réseaux complexes (Relationships Dynamics in Lorenzaccio: Modeling through Complex Networks)." In: *Journées d'Informatique Théâtrale*. 2024. URL: https://ictt.univ-avignon.fr/jit2024en/.

[9] A. Amalvy, M. Janickyj, S. Mannion, P. MacCarron, and V. Labatut. "Interconnected Kingdoms: Comparing 'A Song of Ice and Fire' Adaptations Across Media Using Complex Networks." In: *Social Network Analysis and Mining* (2024). DOI: 10.1007/s13278-024-01365-z.

[10] A. Amalvy and V. Labatut. *Annotation Guidelines for Corpus Novelties: Part 1 — Named Entity Recognition*. Tech. rep. Avignon Université, 2024. URL: https://hal.science/hal-04715338.

[11] A. Amalvy and V. Labatut. *Annotation Guidelines for Corpus Novelties: Part 2 — Alias Resolution*. Tech. rep. Avignon Université, 2024. URL: https://hal.science/hal-04715341.

[12] A. Amalvy, V. Labatut, and R. Dufour. "Data Augmentation for Robust Character Detection in Fantasy Novels." In: *Workshop on Computational Methods in the Humanities (COMHUM)*. 2022, p. 03617722. URL: https://hal.archives-ouvertes.fr/hal-03617722.

[13] A. Amalvy, V. Labatut, and R. Dufour. "Data Augmentation for Robust Character Detection in Fantasy Novels." In: *Workshop on Computational Methods in the Humanities 2022*. 2022, 23–32. URL: https://ceur-ws.org/Vol-3602/paper2.pdf.

[14] A. Amalvy, V. Labatut, and R. Dufour. "Remplacement de mentions pour l'adaptation d'un corpus de reconnaissance d'entités nommées à un domaine cible (Mention Replacement for the Adaptation of a Named Entity Recognition Dataset to a Target Domain)." In: *29e Conférence sur le Traitement Automatique des Langues Naturelles*. Vol. 1. 2022, pp. 197–204. URL: http://talnarchives.atala.org/TALN/TALN-2022/8782.pdf.

[15] A. Amalvy, V. Labatut, and R. Dufour. *Grimbert*. 2023. URL: https://github.com/CompNet/Grimbert.

[16] A. Amalvy, V. Labatut, and R. Dufour. "Learning to Rank Context for Named Entity Recognition Using a Synthetic Dataset." In: *Conference on Empirical Methods in Natural Language Processing*. 2023, pp. 10372–10382. DOI: 10.18653/v1/2023.emnlp-main.642.

[17] A. Amalvy, V. Labatut, and R. Dufour. "The Role of Global and Local Context in Named Entity Recognition." In: *61st Annual Meeting of the Association for Computational Linguistics*. Vol. 2. 2023, pp. 714–722. DOI: 10.18653/v1/2023.acl-short.62.

[18] A. Amalvy, V. Labatut, and R. Dufour. *Tibert*. 2023. URL: https://github.com/CompNet/Tibert.

[19] A. Amalvy, V. Labatut, and R. Dufour. "Apprendre à classer le contexte pour la reconnaissance d'entités nommées en utilisant un jeu de données synthétique (Learning to Rank Context for Named Entity Recognition Using a Synthetic Dataset)." In: *COnférence en Recherche d'InformAtion*. 2024. URL: http://coria.asso-aria.org/2024/articles/abstract_10/main.pdf.

[20] A. Amalvy, V. Labatut, and R. Dufour. "Renard: A Modular Pipeline for Extracting Character Networks from Narrative Texts." In: *Journal of Open Source Software* 9.98 (2024), p. 6574. DOI: 10.21105/joss.06574.

[21] A. Amalvy, V. Labatut, and R. Dufour. "The Role of Natural Language Processing Tasks in Automatic Literary Character Network Construction." In: *31st International Conference on Computational Linguistics*. 2024.

[22] J. M. Anthonisse. *The rush in a directed graph*. Tech. rep. Stichting Mathematisch Centrum, 1971.

[23] M. C. Ardanuy and C. Sporleder. "Structure-based Clustering of Novels." In: *3rd Workshop on Computational Linguistics for Literature*. 2014, pp. 31–39. URL: http://www.aclweb.org/anthology/W14-0905.

[24] M. C. Ardanuy and C. Sporleder. "Clustering of novels represented as social networks." In: *Linguistic Issues in Language Technology* 12 (2015). URL: https://aclanthology.org/2015.lilt-12.4/.

[25] T. Aynaud and J.-L. Guillaume. "Static community detection algorithms for evolving networks." In: *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*. 2010, pp. 513–519. URL: https://ieeexplore.ieee.org/document/5520221.

[26] A. Bagga and B. Baldwin. "Algorithms for Scoring Coreference Chains." In: *1st International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*. 1998, pp. 563–566. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ccdacc60d9d68dfc1f94e7c68bd56646c000e4ab.

[27] P. Bajaj et al. "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset." In: *arXiv* cs.CL (2018), p. 1611.09268. URL: https://arxiv.org/abs/1611.09268.

[28] D. Balasuriya, N. Ringland, J. Nothman, T. Murphy, and J. R. Curran. "Named Entity Recognition in Wikipedia." In: *Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*. 2009, pp. 10–18. URL: https://aclanthology.org/W09-3302.

[29] D. Bamman, O. Lewke, and A. Mansoor. "An Annotated Dataset of Coreference in English Literature." In: *12th Language Resources and Evaluation Conference*. 2020, pp. 44–54. URL: https://aclanthology.org/2020.lrec-1.6.

[30] D. Bamman, S. Popat, and S. Shen. "An annotated dataset of literary entities." In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Vol. 1. 2019, pp. 2138–2144. DOI: 10.18653/v1/N19-1220.

[31] D. Bamman, T. Underwood, and N. A. Smith. "A Bayesian Mixed Effects Model of Literary Character." In: *52nd Annual Meeting of the Association for Computational Linguistics*. Vol. 1. 2014, pp. 370–379. DOI: 10.3115/v1/P14-1035.

[32] N. Bansal, A. Blum, and S. Chawla. "Correlation Clustering." In: *43rd Annual IEEE Symposium on Foundations of Computer Science*. 2002, pp. 238–247. DOI: 10.1109/SFCS.2002.1181947.

[33] A.-L. Barabási and R. Albert. "Emergence of scaling in random networks." In: *science* 286.5439 (1999), pp. 509–512. DOI: 10.1126/science.286.5439.509.

[34] F. Barth, E. Kim, S. Murr, and R. Klinger. "A Reporting Tool for Relational Visualization and Analysis of Character Mentions in Literature." In: *Digital Humanities im deutschsprachigen Raum*. 2018. DOI: 10.5281/zenodo.4622447.

[35] D. S. Bassett. "Network neuroscience." In: *Nature neuroscience* 20.3 (2017), pp. 353–364. DOI: 10.1038/nn.4502.

[36] A. Baudin, L. Tabourier, and C. Magnien. "LSCPM: communities in massive real-world Link Streams by Clique Percolation Method." In: *30th International Symposium on Temporal Representation and Reasoning*. 2023. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TIME.2023.3.

[37] I. Beltagy, M. E. Peters, and A. Cohan. "Longformer: The Long-Document Transformer." In: *arXiv* cs.CL (2020), p. 2004.05150. URL: https://arxiv.org/abs/2004.05150.

[38] S. Bird, E. Loper, and E. Klein. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[39] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. "Fast unfolding of communities in large networks." In: *Journal of Statistical Mechanics* P10008 (2008), pp. 1–12. DOI: 10.1088/1742-5468/2008/10/P10008.

[40] B. Bohnet, C. Alberti, and M. Collins. "Coreference Resolution through a seq2seq Transition-Based System." In: *Transactions of the Association for Computational Linguistics* 11 (2023), pp. 212–226. DOI: 10.1162/tacl_a_00543.

[41] A. Bolioli, M. Casu, M. Lana, and R. Roda. "Exploring the Betrothed Lovers." In: *Workshop on Computational Models of Narrative*. 2013, pp. 30–35. DOI: 10.4230/OASIcs.CMN.2013.30.

[42] P. Bonacich. "Factoring and weighting approaches to status scores and clique identification." In: *The Journal of Mathematical Sociology* 2.1 (1972), pp. 113–120. DOI: 10.1080/0022250X.1972.9989806.

[43] P. Bonacich. "Power and Centrality: A Family of Measures." In: *American Journal of Sociology* 92.5 (1987), pp. 1170–1182. URL: http://www.jstor.org/stable/2780000.

[44] X. Bost. "A storytelling machine ? : automatic video summarization : the case of TV series." PhD Thesis. Université d'Avignon et des Pays de Vaucluse, Laboratoire Informatique d'Avignon, 2016. URL: https://tel.archives-ouvertes.fr/tel-01637270.

[45] X. Bost, S. Gueye, V. Labatut, M. Larson, G. Linarès, D. Malinas, and R. Roth. "Remembering winter was coming: Character-oriented video summaries of TV series." In: *Multimedia Tools and Applications* 78 (2019), pp. 35373–35399. DOI: 10.1007/s11042-019-07969-4.

[46] T. Brown et al. "Language Models are Few-Shot Learners." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[47] R. J. G. B. M. Campello. "Density-Based Clustering Based on Hierarchical Density Estimates." In: *Advances in Knowledge Discovery and Data Mining*. 2013, pp. 160–172. DOI: 10.1007/978-3-642-37456-2_14.

[48] D. Cartwright and F. Harary. "Structural balance: A generalization of Heider's theory." In: *Psychological Review 63* (1956), pp. 277–293. DOI: 10.1037/h0046049.

[49] H. Chen, Z. Fan, H. Lu, A. Yuille, and S. Rong. "PreCo: A Large-scale Dataset in Preschool Vocabulary for Coreference Resolution." In: *Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii. 2018, pp. 172–181. DOI: 10.18653/v1/D18-1016.

[50] S. Chen, G. Aguilar, L. Neves, and T. Solorio. "Data Augmentation for Cross-Domain Named Entity Recognition." In: *Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 5346–5356. DOI: 10.18653/v1/2021.emnlp-main.434.

[51] W. Chiang et al. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

[52] J. Cohen. "A coefficient of agreement for nominal scales." In: *Educational and psychological measurement* 20.1 (1960), pp. 37–46.

[53] M. Condello, R. Harrison, J. Isasi, A. Kinnaman, and A. Kumari. "A Methodology for Character Networks at the Macroanalytical Level." In: *Digital Humanities Forum*. 2014. URL: https://idrh.drupal.ku.edu/sites/idrh.ku.edu/files/files/dhforum2014/abstract-Condello-Character-Networks.pdf.

[54] L. F. Costa, O. Jr. N. Oliveira, G. Travieso, F. A. Rodrigues, P. R. V. Boas, L. Antiqueira, M. P. Viana, and L. E. C. Rocha. "Analyzing and modeling real-world phenomena with complex networks: a survey of applications." In: *Advances in Physics* 60.3 (2011), pp. 329–412. DOI: 10.1080/00018732.2011.572452.

[55] L. d. F. Costa, F. A. Rodrigues, G. Travieso, and P. R. Villas Boas. "Characterization of complex networks: A survey of measurements." In: *Advances in Physics* 56.1 (2007), pp. 167–242. DOI: 10.1080/00018730601170527.

[56] C. Cuesta-Lazaro, A. Prasad, and T. Wood. "What does the sea say to the shore ? A BERT based DST style approach for speaker to dialogue attribution in novels." In: *60th Annual Meeting of the Association for Computational Linguistics*. 2022. URL: https://www.amazon.science/publications/what-does-the-sea-say-to-the-shore-a-bert-based-dst-style-approach-for-speaker-to-dialogue-attribution-in-novels.

[57] C. Cui et al. "A Survey on Multimodal Large Language Models for Autonomous Driving." In: *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. 2024, pp. 958–979. DOI: 10.1109/WACVW60836.2024.00106.

[58] X. Dai and H. Adel. "An Analysis of Simple Data Augmentation for Named Entity Recognition." In: *International Conference on Computational Linguistics*. 2020, pp. 3861–3867. DOI: 10.18653/v1/2020.coling-main.343.

[59] N. Dekker, T. Kuhn, and M. van Erp. "Evaluating named entity recognition tools for extracting social networks from novels." In: *PeerJ Computer Science* 5 (2019), e189. DOI: 10.7717/peerj-cs.189.

[60] J. Devlin, M. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Vol. 1. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[61] B. Ding, L. Liu, L. Bing, C. Kruengkrai, T. H. Nguyen, S. Joty, L. Si, and C. Miao. "DAGA: Data Augmentation with a Generation Approach for Low-resource Tagging Tasks." In: *Conference on Empirical Methods in Natural Language Processing*. 2020, pp. 6045–6057. DOI: 10.18653/v1/2020.emnlp-main.488.

[62] T. T. H. Do, Q. H. B. Tran, and Q. D. Tran. "Movie indexing and summarization using social network techniques." In: *Vietnam Journal of Computer Science* 5.2 (2018), pp. 157–164. DOI: 10.1007/s40595-018-0111-2.

[63] V. Dobrovolskii. "Word-Level Coreference Resolution." In: *Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 7670–7675. DOI: 10.18653/v1/2021.emnlp-main.605.

[64] N. Dugué, V. Labatut, and A. Perez. "A community role approach to assess social capitalists visibility in the Twitter network." In: *Social Network Analysis and Mining* 5.26 (2015). DOI: 10.1007/s13278-015-0266-0.

[65] N. Durandard, V. A. Tran, G. Michel, and E. Epure. "Automatic Annotation of Direct Speech in Written French Narratives." In: *61st Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Vol. 1. 2023, pp. 7129–7147. DOI: 10.18653/v1/2023.acl-long.393.

[66] M. Ehrmann. "Les Entités Nommées, de la linguistique au TAL : Statut théorique et méthodes de désambiguïsation." PhD Thesis. Université Paris Diderot, 2008. URL: https://theses.hal.science/tel-01639190.

[67] J. L. Elman. "Finding Structure in Time." In: *Cognitive Science* 14.2 (1990), pp. 179–211. DOI: 10.1016/0364-0213(90)90002-E.

[68] D. K. Elson. "Modeling Narrative Discourse." PhD thesis. Colombia University, Dec. 2012. URL: http://www.cs.columbia.edu/~delson/pubs/Modeling-Narrative-Discourse_Elson_R4.pdf.

[69] D. Elson, N. Dames, and K. McKeown. "Extracting Social Networks from Literary Fiction." In: *48th Annual Meeting of the Association for Computational Linguistics*. Ed. by Jan Hajič, Sandra Carberry, Stephen Clark, and Joakim Nivre. 2010, pp. 138–147. URL: https://aclanthology.org/P10-1015.

[70] D. Elson and K. McKeown. "Automatic Attribution of Quoted Speech in Literary Narrative." In: *AAAI Conference on Artificial Intelligence*. 2010. URL: http://www.cs.columbia.edu/~delson/pubs/AAAI10-ElsonMcKeown.pdf.

[71] M. Falk. "Making Connections: Network Analysis, the Bildungsroman and the World of The Absentee." In: *Journal of Language, Literature and Culture* 63.2-3 (2016), pp. 107–122. DOI: 10.1080/20512856.2016.1244909.

[72] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy. "A Survey of Data Augmentation Approaches for NLP." In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP*. 2021, pp. 968–988. DOI: 10.18653/v1/2021.findings-acl.84.

[73] F. Fischer, I. Börner, M. Göbel, A. Hechtl, C. Kittel, C. Milling, and P. Trilcke. "Programmable Corpora: Introducing DraCor, an Infrastructure for the Research on European Drama." In: *Digital Humanities 2019*. 2019. DOI: 10.5281/zenodo.4284002.

[74] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. "Named Entity Recognition through Classifier Combination." In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 168–171. URL: https://aclanthology.org/W03-0425.

[75] S. Fortunato. "Community Detection in Graphs." In: *Physics Reports* 486 (3-5 2010), pp. 75–174. DOI: 10.1016/j.physrep.2009.11.002.

[76] L. C. Freeman. "A set of measures of centrality based on betweenness." In: *Sociometry* 40.1 (1977), pp. 35–41.

[77] L. C. Freeman. "Centrality in social networks: conceptual clarification." In: *Social Networks* 1.3 (1978), pp. 215–239. DOI: 10.1016/0378-8733(78)90021-7.

[78] K. Fukushima. "A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." In: *Biological Cybernetics* 36 (1980), pp. 193–202. DOI: 10.1007/BF00344251.

[79] T. Gessey-Jones, C. Connaughton, R. Dunbar, R. Kenna, P. Mac Carron, C. O'Conchobhair, and J. Yose. "Narrative structure of A Song of Ice and Fire creates a fictional world with realistic measures of social complexity." In: *Proceedings of the National Academy of Sciences* 117.46 (2020), pp. 28582–28588. DOI: 10.1073/pnas.2006465117.

[80] A. Ghaddar and P. Langlais. "WikiCoref: An English Coreference-annotated Corpus of Wikipedia Articles." In: *10th International Conference on Language Resources and Evaluation*. 2016, pp. 136–142. URL: https://aclanthology.org/L16-1021.

[81] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. "Neural message passing for Quantum chemistry." In: *34th International Conference on Machine Learning*. Vol. 70. 2017, 1263–1272. DOI: 10.1063/1.464913.

[82] K. R. Glass and S. Bangay. "A naïve, salience-based method for speaker identification in fiction books." In: *18th Annual Symposium of the Pattern Recognition Association of South Africa*. 2007, pp. 1–6. URL: https://www.cs.ru.ac.za/research/g05g1909/papers/prasa07b.pdf.

[83] S. Grayson, K. Wade, G. Meaney, and D. Greene. "The Sense and Sensibility of Different Sliding Windows in Constructing Co-occurrence Networks from Literature." In: *Computational History and Data-Driven Humanities*. 2016, pp. 65–77. DOI: 10.1007/978-3-319-46224-0_7.

[84] M. Grootendorst. "BERTopic: Neural topic modeling with a class-based TF-IDF procedure." In: *arXiv* cs.CL (2022), p. 2203.05794. URL: https://arxiv.org/abs/2203.05794.

[85] R. Guimerà and L. A. N. Amaral. "Functional cartography of complex metabolic networks." In: *Nature* 433 (2005), pp. 895–900. DOI: 10.1038/nature03288.

[86] Q. Guo, X. Hu, Y. Zhang, X. Qiu, and Z. Zhang. "Dual Cache for Long Document Neural Coreference Resolution." In: *61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. 2023, pp. 15272–15285. DOI: 10.18653/v1/2023.acl-long.851.

[87] T. Gupta, H. O. Hatzel, and C. Biemann. "Coreference in Long Documents using Hierarchical Entity Merging." In: *Google Scholar* (2024). URL: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2024-gupta-et-al-sighum.pdf.

[88] 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." In: *58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 8342–8360. DOI: 10.18653/v1/2020.acl-main.740.

[89] A. A. Hagbert, D. A. Schult, and P. J. Swart. "Exploring network structure, dynamics and function using NetworkX." In: *7th Python in Science Conference (SciPy2008)*. 2008. URL: https://www.osti.gov/biblio/960616.

[90] H. He, D. Barbosa, and G. Kondrak. "Identification of Speakers in Novels." In: *51st Annual Meeting of the Association for Computational Linguistics*. Vol. 1. 2013, pp. 1312–1320. URL: https://aclanthology.org/P13-1129.

[91] L. Hettinger, M. Becker, I. Reger, F. Jannidis, and A. Hotho. "Genre Classification on German Novels." In: *26th International Workshop on Database and Expert Systems Applications (DEXA)*. 2015, pp. 249–253. DOI: 10.1109/DEXA.2015.62.

[92] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory." In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[93] A. J. Holanda, M. Matias, S. M. S. P Ferreira, G. M. L. Benevides, and O. Kinouchi. "Character Networks and Book Genre Classification." In: *International Journal of Modern Physics* 30.08 (2019). DOI: 10.1016/j.physrep.2012.03.001.

[94] P. Holme and J. Saramäki. "Temporal networks." In: *Physics Reports* 519.3 (2012). Temporal Networks, pp. 97–125. DOI: 10.1016/j.physrep.2012.03.001.

[95] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. *spaCy: Industrial-strength Natural Language Processing in Python*. 2020. DOI: 10.5281/zenodo.1212303.

[96] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang. "GraphMAE: Self-Supervised Masked Graph Autoencoders." In: *28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, 594–604. ISBN: 9781450393850. DOI: 10.1145/3534678.3539321.

[97] L. Hutcheon. *A Theory of Adaptation*. New York: Routledge, 2006. DOI: 10.4324/9780203957721.

[98] C. Hutto and E. Gilbert. "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text." In: *International AAAI Conference on Web and Social Media*. Vol. 8. 1. 2014, pp. 216–225. DOI: 10.1609/icwsm.v8i1.14550.

[99] L. Jahan, R. Mittal, W. V. Yarlott, and M. Finlayson. "A Straightforward Approach to Narratologically Grounded Character Identification." In: *28th International Conference on Computational Linguistics*. 2020, pp. 6089–6100. DOI: 10.18653/v1/2020.coling-main.536.

[100] P. Jayannavar, A. Agarwal, M. Ju, and O. Rambow. "Validating Literary Theories Using Automatic Social Network Extraction." In: *4th Workshop on Computational Linguistics for Literature*. Ed. by Anna Feldman, Anna Kazantseva, Stan Szpakowicz, and Corina Koolen. 2015, pp. 32–41. DOI: 10.3115/v1/W15-0704.

[101] M. Joshi, D. Chen, Y. Liu, D. Weld S., L. Zettlemoyer, and O. Levy. "SpanBERT: Improving Pre-training by Representing and Predicting Spans." In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 64–77. URL: https://transacl.org/ojs/index.php/tacl/article/view/1853.

[102] M. Joshi, O. Levy, L. Zettlemoyer, and D. Weld. "BERT for Coreference Resolution: Baselines and Analysis." In: *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 2019, pp. 5803–5808. DOI: 10.18653/v1/D19-1588.

[103] R. Kenna and P. Mac Carron. "Maths meets myths: Network investigations of ancient narratives." In: *Journal of Physics: Conference Series* 681 (2016), p. 012002. DOI: 10.1088/1742-6596/681/1/012002.

[104] R. Kenna and P. Mac Carron. "A Networks Approach to Mythological Epics." In: *Maths Meets Myths: Quantitative Approaches to Ancient Narratives*. 2017, pp. 21–43. DOI: 10.1007/978-3-319-39445-9_3.

[105] I. Keraghel, S. Morbieu, and M. Nadif. "A survey on recent advances in named entity recognition." In: *arXiv* cs.CL (2024), p. 2401.10825. URL: https://arxiv.org/abs/2401.10825.

[106] D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski. "Centrality Indices." In: *Lecture Notes in Computer Sciences* 3418 (2005), pp. 16–61. DOI: 10.1007/978-3-540-31955-9_3.

[107] M. Krug. "Techniques for the Automatic Extraction of Character Networks in German Historic Novels." PhD thesis. Julius Maximilians University Würzburg, 2020. URL: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2024-gupta-et-al-sighum.pdf.

[108] D. Kydros and A. Anastasiadis. "Social network analysis in literature. The case of The Great Eastern by A. Embirikos." In: *5th European Congress of Modern Greek Studies of the European Society of Modern Greek Studies*. 2015, pp. 681–702. URL: https://www.researchgate.net/profile/Dimitrios_Kydros/publication/273357563_Social_network_analysis_in_literature._The_case_of_The_Great_Eastern_by_A._Embirikos/links/54ff172f0cf2672e22419bd6.pdf.

[109] V. Labatut. "Complex Network Analysis of a Graphic Novel: The Case of the Bande Dessinée Thorgal." In: *Advances in Complex Systems* 25.5&6 (2022), p. 2240003. DOI: 10.1142/S0219525922400033.

[110] V. Labatut and X. Bost. "Extraction and Analysis of Fictional Character Networks : A Survey." In: *ACM Computing Surveys* 52 (2019), p. 89. DOI: 10.1145/3344548.

[111] F. Landragin. "Description, modélisation et détection automatique des chaînes de référence." In: *Bulletin de l'Association Française pour l'Intelligence Artificielle* 92 (2016), pp. 11–15. URL: https://hal.archives-ouvertes.fr/hal-01347949.

[112] Lattice. *fr-litbank: A French Litbank Corpus*. 2022. URL: https://github.com/lattice-8094/fr-litbank.

[113] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. "End-to-end Neural Coreference Resolution." In: *Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 188–197. DOI: 10.18653/v1/D17-1018.

[114] K. Lee, L. He, and L. Zettlemoyer. "Higher-Order Coreference Resolution with Coarse-to-Fine Inference." In: *Conference of the North American Chapter of the Association for Computational Linguistics*. Vol. 2. 2018, pp. 687–692. DOI: 10.18653/v1/N18-2108.

[115] O.-J. Lee and J. J. Jung. "Modeling affective character network for story analytics." In: *Future Generation Computer Systems* 92 (2019), pp. 458–478. DOI: 10.1016/j.future.2018.01.030.

[116] J. Li, R. Wu, W. Sun, L. Chen, S. Tian, L. Zhu, C. Meng, Z. Zheng, and W. Wang. "What's Behind the Mask: Understanding Masked Graph Modeling for Graph Autoencoders." In: *29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, 1268–1279. DOI: 10.1145/3580305.3599546.

[117] C. Lignos and M. Kamyab. "If You Build Your Own NER Scorer, Non-replicable Results Will Come." In: *1st Workshop on Insights from Negative Results in NLP*. 2020, pp. 94–99. DOI: 10.18653/v1/2020.insights-1.15.

[118] R. Liu, R. Mao, A. T. Luu, and E. Cambria. "A brief survey on recent advances in coreference resolution." In: *Artificial Intelligence Review* 56 (2023), pp. 14439–14481. DOI: 10.1007/s10462-023-10506-3.

[119] T. Liu, Y. E. Jiang, N. Monath, R. Cotterell, and M. Sachan. "Autoregressive Structured Prediction with Language Models." In: *Findings of the Association for Computational Linguistics: EMNLP*. 2022, pp. 993–1005. DOI: 10.18653/v1/2022.findings-emnlp.70.

[120] T. Liu, J. Yao, and C. Lin. "Towards Improving Neural Named Entity Recognition with Gazetteers." In: *57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 5301–5307. DOI: 10.18653/v1/P19-1524.

[121] S. P. Lloyd. "Least squares quantization in PCM." In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[122] J. Long. "Large Language Model Guided Tree-of-Thought." In: *arXiv* cs.AI (2023), p. 2305.08291. URL: https://arxiv.org/abs/2305.08291.

[123] G. Luo, X. Huang, C. Lin, and Z. Nie. "Joint Entity Recognition and Disambiguation." In: *Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 879–888. DOI: 10.18653/v1/D15-1104.

[124] X. Luo. "On Coreference Resolution Performance Metrics." In: *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. 2005, pp. 25–32. URL: https://aclanthology.org/H05-1004.

[125] J. Luoma and S. Pyysalo. "Exploring Cross-sentence Contexts for Named Entity Recognition with BERT." In: *28th International Conference on Computational Linguistics*. 2020, pp. 904–914. DOI: 10.18653/v1/2020.coling-main.78.

[126] P. Mac Carron. "A Network Theoretic Approach to Comparative Mythology." PhD thesis. Applied Mathematics Research Centre, Coventry University, 2014. URL: https://curve.coventry.ac.uk/open/file/9bfc043d-497e-4217-82ab-e19963b53290/1/maccarroncomb.pdf.

[127] P. Mac Carron and R. Kenna. "Universal Properties of Mythological Networks." In: *Europhysics Letters* 99.2 (2012), p. 28002. DOI: 10.1209/0295-5075/99/28002.

[128] P. Mac Carron and R. Kenna. "Network analysis of the Íslendinga sögur - the Sagas of Icelanders." In: *European Physical Journal B* 86 (2013), p. 407. DOI: 10.1140/epjb/e2013-40583-3.

[129] P. Mac Carron and R. Kenna. "A quantitative approach to comparative mythology." In: *Cosmos* 14 (2014), pp. 103–117. URL: https://ora.ox.ac.uk/objects/uuid:43275d59-4db2-49db-b26a-96f21378cfa8.

[130] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. DOI: 10.1017/CBO9780511809071.

[131] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit." In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

[132] H. Mao, Z. Chen, W. Tang, J. Zhao, Y. Ma, T. Zhao, N. Shah, M. Galkin, and J. Tang. "Position: Graph Foundation Models are Already Here." In: *arXiv* cs.LG (2024), p. 2402.02216. URL: https://arxiv.org/abs/2402.02216.

[133] L. Martin, B. Muller, S. Ortiz, J. Pedro, Y. Dupont, L. Romary, É. de la Clergerie, D. Seddah, and B. Sagot. "CamemBERT: a Tasty French Language Model." In: *58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 7203–7219. URL: https://www.aclweb.org/anthology/2020.acl-main.645.

[134] L. McInnes, J. Healy, and J. Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." In: *arXiv* stat.ML (2020), p. 1802.03426. URL: https://arxiv.org/abs/1802.03426.

[135] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space." In: *arXiv* cs.CL (2013), p. 1301.3781. DOI: 10.48550/ARXIV.1301.3781.

[136] S. Min and J. Park. "Network Science and Narratives: Basic Model and Application to Victor Hugo's Les Misérables." In: *Studies in Computational Intelligence* 644 (2016), pp. 257–265. DOI: 10.1007/978-3-319-30569-1_19.

[137] N. S. Moosavi and M. Strube. "Which Coreference Evaluation Metric Do You Trust? A Proposal for a Link-based Entity Aware Metric." In: *54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Katrin Erk and Noah A. Smith. 2016, pp. 632–642. DOI: 10.18653/v1/P16-1060.

[138] F. Moretti. "Network Theory, Plot Analysis." In: *Stanford Literary Lab* 2 (2011). URL: https://litlab.stanford.edu/LiteraryLabPamphlet2.pdf.

[139] N. Muennighoff et al. "Crosslingual Generalization through Multitask Finetuning." In: *61st Annual Meeting of the Association for Computational Linguistics*. Vol. 1. 2023, pp. 15991–16111. DOI: 10.18653/v1/2023.acl-long.891.

[140] S. Muhuri, S. Chakraborty, and S. N. Chakraborty. "Extracting Social Network and Character Categorization From Bengali Literature." In: *IEEE Transactions on Computational Social Systems* in press (2018). DOI: 10.1109/TCSS.2018.2798699.

[141] G. Muzny, M. Fang, A. Chang, and D. Jurafsky. "A Two-stage Sieve Approach for Quote Attribution." In: *15th Conference of the European Chapter of the Association for Computational Linguistics*. Vol. 1. 2017, pp. 460–470. URL: https://aclanthology.org/E17-1044.

[142] C. Métrailler. *Charnetto*. 2023. URL: https://gitlab.com/maned_wolf/charnetto.

[143] D. Nadeau and S. Sekine. "A survey of named entity recognition and classification." In: *Lingvisticae Investigationes* 30.1 (2007), pp. 3–26. DOI: 10.1075/li.30.1.03nad.

[144] H. Nakayama. *seqeval: A Python framework for sequence labeling evaluation*. 2018. URL: https://github.com/chakki-works/seqeval.

[145] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. "graph2vec: Learning Distributed Representations of Graphs." In: *arXiv* cs.AI (2017), p. 1707.05005. URL: https://arxiv.org/abs/1707.05005.

[146] M. E. J. Newman. "The structure and function of complex networks." In: *SIAM review* 45.2 (2003), pp. 167–256. DOI: 10.1137/S003614450342480.

[147] R. Nogueira and K. Cho. "Passage Re-ranking with BERT." In: *arXiv* cs.IR (2020), p. 1901.04085. URL: https://arxiv.org/abs/1901.04085.

[148] R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin. "Document Ranking with a Pretrained Sequence-to-Sequence Model." In: *Findings of the Association for Computational Linguistics: EMNLP*. 2020, pp. 708–718. DOI: 10.18653/v1/2020.findings-emnlp.63.

[149] T. O'Keefe, S. Pareti, J. R. Curran, I. Koprinska, M. K. Honnibal, S. Pareti, J. R. Curran, I. Koprinska, and M. Honnibal. "A Sequence Labelling Approach to Quote Attribution." In: *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 2012, pp. 790–799. URL: https://aclanthology.org/D12-1072.

[150] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. "Uncovering the overlapping community structure of complex networks in nature and society." In: *Nature* 435.7043 (2005), pp. 814–818. DOI: 10.1038/nature03607.

[151] G.-M. Park, S.-H. Kim, and H.-G. Cho. "Structural analysis on social network constructed from characters in literature texts." In: *Journal of Computers* 8.9 (2013), pp. 2442–2447. URL: http://www.jcomputers.us/vol8/jcp0809-38.pdf.

[152] G.-M. Park, S.-H. Kim, H.-R. Hwang, and H.-G. Cho. "Complex System Analysis of Social Networks Extracted from Literary Fictions." In: *International Journal of Machine Learning and Computing* 3.1 (2013), pp. 107–111. DOI: 10.7763/IJMLC.2013.V3.282.

[153] S.-B. Park, K.-J. Oh, and G.-S. Jo. "Social network analysis in a movie using CharacterNet." In: *Multimedia Tools and Applications* 59.2 (2011), pp. 601–627. DOI: 10.1007/s11042-011-0725-1.

[154] Z. Patel, K. El-Refai, J. Pei, and T. Li. "SWAG: Storytelling With Action Guidance." In: *arXiv* cs.CL (2024), p. 2402.03483. URL: https://arxiv.org/abs/2402.03483.

[155] T. Pial, S. Salim, C. Pethe, A. Kim, and S. Skiena. "Analyzing Film Adaptation through Narrative Alignment." In: *arXiv* cs.CL (2023), p. 2311.04020. URL: https://arxiv.org/abs/2311.04020.

[156] T. Pial and S. Skiena. "GNAT: A General Narrative Alignment Tool." In: *Conference on Empirical Methods in Natural Language Processing*. 2023, pp. 14636–14652. DOI: 10.18653/v1/2023.emnlp-main.904.

[157] M. Poesio, R. Stuckardt, and Y. Versley. *Anaphora resolution: Algorithms, resources, and applications*. Springer, 2016.

[158] J. D. Power, B. L. Schlaggar, C. N. Lessov-Schlaggar, and S. E. Petersen. "Evidence for hubs in human functional brain networks." In: *Neuron* 79.4 (2013), pp. 798–813. DOI: 10.1016/j.neuron.2013.07.035.

[159] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang. "CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes." In: *Joint Conference on EMNLP and CoNLL*. 2012, pp. 1–40. URL: https://aclanthology.org/W12-4501.

[160] S. D. Prado, S. R. Dahmen, A. L. C. Bazzan, P. Mac Carron, and R. Kenna. "Temporal Network Analysis of Literary Texts." In: *Advances in Complex Systems* 19.3 (2016), p. 1650005. DOI: 10.1142/S0219525916500053.

[161] L. Ramshaw and M. Marcus. "Text Chunking using Transformation-Based Learning." In: *Third Workshop on Very Large Corpora*. 1995. URL: https://aclanthology.org/W95-0107.

[162] L. F. Rau. "Extracting company names from text." In: *7th Conference on Artificial Intelligence Application*. 1991, pp. 29–30. DOI: 10.1109/CAIA.1991.120841.

[163] M. Recasens and E. Hovy. "BLANC: Implementing the Rand index for coreference evaluation." In: *Natural Language Engineering* 17 (2011), pp. 485–510. DOI: 10.1017/S135132491000029X.

[164] N. Reimers and I. Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410.

[165] L. Renze, Z. Kai, and Y. Wenpeng. "Is Prompt All You Need? No. A Comprehensive and Broader View of Instruction Learning." In: *arXiv* cs.CL (2023), p. 2303.10475. URL: https://arxiv.org/abs/2303.10475.

[166] S. E. W. Robertson. "Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval." In: *SIGIR '94*. Ed. by Bruce W. Croft and C. J. van Rijsbergen. 1994, pp. 232–241. ISBN: 978-1-4471-2099-5.

[167] Y. Rochat. "Character Networks and Centrality." PhD thesis. Université de Lausanne, Dec. 2014. URL: https://serval.unil.ch/resource/serval:BIB_663137B68131.P001/REF.pdf.

[168] Y. Rochat. "Character network analysis of Émile Zola's Les Rougon-Macquart." In: *Digital Humanities*. 2015. URL: https://infoscience.epfl.ch/record/210573?ln=en.

[169] Y. Rochat and F. Kaplan. "Analyse de réseaux sur les personnages des Confessions de Jean-Jacques Rousseau." In: *Cahiers du numérique* 10.3 (2014), pp. 109–133. DOI: 10.3166/LCN.10.3.109-133.

[170] Y. Rochat and M. Triclot. "Les réseaux de personnages de science-fiction : échantillons de lectures intermédiaires." In: *ReS Futurae* 10 (2017), p. 1183. DOI: 10.4000/resf.1183.

[171] A. Rueda, E. Alvarez-Mellado, and C. Lignos. "CoNLL#: Fine-grained Error Analysis and a Corrected Test Set for CoNLL-03 English." In: *Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING)*. Ed. by Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue. 2024, pp. 3718–3728. URL: https://aclanthology.org/2024.lrec-main.330.

[172] M. Ružička. "Anwendung mathematisch-statistiker Methoden in Geobotanik (Synthetische Bearbeitung von Aufnahmen)." In: *Biologia* 13 (1958), pp. 647–661.

[173] G. A. Sack. "Character Networks for Narrative Generation." In: *8th Artificial Intelligence and Interactive Digital Entertainment Conference - Intelligent Narrative Technologies Workshop*. 2012, pp. 38–43. URL: http://www.aaai.org/ocs/index.php/AIIDE/AIIDE12/paper/view/5550.

[174] G. A. Sack. "Character networks for narrative generation: Structural balance theory and the emergence of proto-narratives." In: *Workshop on Computational Models of Narrative*. 2013. DOI: 10.4230/OASIcs.CMN.2013.183.

[175] G. A. Sack. "Character Networks for Narrative Generation: Structural Balance Theory and the Emergence of Proto-Narratives." In: *Complexity and the Human Experience - Modeling Complexity in the Humanities and Social Sciences*. Pan Stanford Publishing - CRC Press, 2014. Chap. 4, pp. 81–104. DOI: 10.4032/9789814463270.

[176] M. Saleh, Y. Esa, and A. Mohamed. "Applications of Complex Network Analysis in Electric Power Systems." In: *Energies* 11.6 (2018). DOI: 10.3390/en11061381.

[177] V. Sanh et al. "Multitask Prompted Training Enables Zero-Shot Task Generalization." In: *10th Internation Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=9Vrb9D0WI4.

[178] A. Saxena and S. Iyengar. "Centrality Measures in Complex Networks: A Survey." In: *arXiv* cs.SI (2020), p. 2011.07190. URL: https://arxiv.org/abs/2011.07190.

[179] J. Seo, G.-M. Park, S.-H. Kim, and H.-G. Cho. "Characteristic Analysis of Social Network Constructed from Literary Fiction." In: *International Conference on Cyberworlds*. 2013, pp. 147–150. DOI: 10.1109/CW.2013.72.

[180] D. Seyler, T. Dembelova, L. Del Corro, J. Hoffart, and G. Weikum. "A Study of the Importance of External Knowledge in the Named Entity Recognition Task." In: *56th Annual Meeting of the Association for Computational Linguistics*. Vol. 2. 2018, pp. 241–246. DOI: 10.18653/v1/P18-2039.

[181] C. Shorten and T. M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning." In: *Journal of Big Data* 6.1 (2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0.

[182] M. Sims and D. Bamman. "Measuring Information Propagation in Literary Social Networks." In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. 2020, pp. 642–652. DOI: 10.18653/v1/2020.emnlp-main.47.

[183] M. Sims, J. H. Park, and D. Bamman. "Literary Event Detection." In: *57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. 2019, pp. 3623–3634. DOI: 10.18653/v1/P19-1353.

[184] T. F. Smith and M. S. Waterman. "Identification of Common Molecular Subsequences." In: *Journal of molecular biology* 147.1 (1981), pp. 195–197.

[185] A. C. Sparavigna. "On Social Networks in Plays and Novels." In: *International Journal of Sciences* 2.10 (2013), pp. 20–25. DOI: 10.18483/ijSci.312.

[186] A. C. Sparavigna and R. Marazzato. "Analysis of a Play by Means of CHAPLIN, the Characters and Places Interaction Network Software." In: *International Journal of Sciences* 4.3 (2015), pp. 60–68. DOI: 10.18483/ijSci.662.

[187] K. Spärck Jones. "A statistical interpretation of term specificity and its application in retrieval." In: *Journal of Documentation* 60 (1972), pp. 493–502. DOI: 10.1108/eb026526.

[188] T. Stanislawek, A. Wróblewska, A. Wójcicka, D. Ziembicki, and P. Biecek. "Named Entity Recognition - Is There a Glass Ceiling?" In: *23rd Conference on Computational Natural Language Learning*. 2019, pp. 624–633. DOI: 10.18653/v1/K19-1058.

[189] Z. Su, L. Xu, J. Xu, J. Li, and M. Huangfu. "SIG: Speaker Identification in Literature via Prompt-Based Generation." In: *AAAI Conference on Artificial Intelligence*. 2024, pp. 19035–19043. URL: https://ojs.aaai.org/index.php/AAAI/article/view/29870/31517.

[190] Y. Tang, J. Li, N. A. H. Haldar, Z. Guan, J. Xu, and C. Liu. "Reliable community search in dynamic networks." In: *VLDB Endowment* 15.11 (2022), 2826–2838. DOI: 10.14778/3551793.3551834.

[191] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. *Stanford Alpaca: An Instruction-following LLaMA model*. 2023. URL: https://github.com/tatsu-lab/stanford_alpaca.

[192] 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." In: *10th International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=qVyeW-grC2k.

[193] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. "Efficient Transformers: A Survey." In: *ACM Computing Survey* 55.6 (2022). DOI: 10.1145/3530811.

[194] E. F. Tjong Kim Sang. "Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition." In: *6th Conference on Natural Language Learning*. 2002. URL: https://aclanthology.org/W02-2024.

[195] E. F. Tjong Kim Sang and F. De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition." In: *7th Conference on Natural Language Learning*. 2003, pp. 142–147. URL: https://aclanthology.org/W03-0419.

[196] S. Toshniwal, S. Wiseman, A. Ettinger, K. Livescu, and K. Gimpel. "Learning to Ignore: Long Document Coreference with Bounded Memory Neural Networks." In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 8519–8526. DOI: 10.18653/v1/2020.emnlp-main.685.

[197] H. Touvron et al. "LLaMA: Open and Efficient Foundation Language Models." In: *arXiv* cs.CL (2023), p. 2302.13971. URL: https://arxiv.org/abs/2302.13971.

[198] Q. D. Tran, D. Hwang, and J. J. Jung. "Movie summarization using characters network analysis." In: *Computational Collective Intelligence*. Vol. 9329. Lecture Notes in Computer Science. Springer, 2015, pp. 390–399. DOI: 10.1007/978-3-319-24069-5_37.

[199] Q. D. Tran, D. Hwang, O-J. Lee, and J. E. Jung. "Exploiting character networks for movie summarization." In: *Multimedia Tools and Applications* 76.8 (2017), pp. 10357–10369. DOI: 10.1007/s11042-016-3633-6.

[200] C.-M. Tsai, L.-W. Kang, C.-W. Lin, and W. Lin. "Scene-Based Movie Summarization Via Role-Community Networks." In: *IEEE Transactions on Circuits and Systems for Video Technology* 23.11 (2013), pp. 1927–1940. DOI: 10.1109/TCSVT.2013.2269186.

[201] G. Urbizu, A. Soraluze, and O. Arregi. "Sequence to Sequence Coreference Resolution." In: *3rd Workshop on Computational Models of Reference, Anaphora and Coreference*. 2020, pp. 39–46. URL: https://aclanthology.org/2020.crac-1.5.

[202] H. Vala, D. Jurgens, A. Piper, and D. Ruths. "Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On The Difficulty of Detecting Characters in Literary Texts." In: *Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 769–774. DOI: 10.18653/v1/D15-1088.

[203] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez N., and I. Polosukhin. "Attention is All you Need." In: *arXiv* cs.CL (2017), p. 1706.03762. URL: https://arxiv.org/abs/1706.03762.

[204] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. "A Model-Theoretic Coreference Scoring Scheme." In: *6th Message Understanding Conference*. 1995. URL: https://aclanthology.org/M95-1005.

[205] K. Vishnubhotla, A. Hammond, and G. Hirst. "The Project Dialogism Novel Corpus: A Dataset for Quotation Attribution in Literary Texts." In: *13th Language Resources and Evaluation Conference*. 2022, pp. 5838–5848. URL: https://aclanthology.org/2022.lrec-1.628.

[206] K. Wang, Y. Ding, and S. C. Han. "Graph neural networks for text classification: a survey." In: *Artificial Intelligence Review* 57 (190 2024). DOI: 10.1007/s10462-024-10808-0.

[207] T. Wang et al. "Weaver: Foundation Models for Creative Writing." In: *arXiv* cs.CL (2024), p. 2401.17268. URL: https://arxiv.org/abs/2401.17268.

[208] X. Wang, Y. Jiang, N Bach, T. Wang, Z. Huang, F Huang, and K. Tu. "Automated Concatenation of Embeddings for Structured Prediction." In: *arXiv* cs.CL (2021), p. 2010.05006. URL: https://arxiv.org/abs/2010.05006.

[209] X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and K. Tu. "Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning." In: *59th Annual Meeting of the Association for Computational Linguistics and 11th International Joint Conference on Natural Language Processing*. Vol. 1. 2021, pp. 1800–1812. DOI: 10.18653/v1/2021.acl-long.142.

[210] Z. Wang, J. Shang, L. Liu, L. Lu, J. Liu, and J. Han. "CrossWeigh: Training Named Entity Tagger from Imperfect Annotations." In: *Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*. 2019, pp. 5154–5163. DOI: 10.18653/v1/D19-1519.

[211] J. Wei, M. Bosma, V. Zhao, K. Guu, A. Wei Yu, B Lester, N. Du, A. M. Dai, and Q. V Le. "Finetuned Language Models Are Zero-Shot Learners." In: *10th International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=gEZrGCozdqR.

[212] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. "Chain-of-thought prompting elicits reasoning in large language models." In: *36th International Conference on Neural Information Processing Systems*. 2024. URL: https://dl.acm.org/doi/10.5555/3600270.3602070.

[213] R. Weischedel, E. Hovy, M. Marcus, M. Palmer, R. Belvin, S. Pradhan, L. Ramshaw, and N. Xue. *OntoNotes: A Large Training Corpus for Enhanced Processing*. 2011. URL: https://www.cs.cmu.edu/~hovy/papers/09OntoNotes-GALEbook.pdf.

[214] C.-Y. Weng, W.-T. Chu, and J.-L. Wu. "Movie analysis based on roles' social network." In: *IEEE International Conference on Multimedia and Expo*. 2007, pp. 1403–1406. DOI: 10.1109/ICME.2007.4284922.

[215] C.-Y. Weng, W.-T. Chu, and J.-L. Wu. "RoleNet: Treat a movie as a small society." In: *International workshop on Workshop on multimedia information retrieval*. 2007, pp. 51–60. DOI: 10.1145/1290082.1290092.

[216] C.-Y. Weng, W.-T. Chu, and J.-L. Wu. "RoleNet: Movie Analysis from the Perspective of Social Networks." In: *IEEE Transactions on Multimedia* 11.2 (2009), pp. 256–271. DOI: 10.1109/TMM.2008.2009684.

[217] Wikipedia. *Game of Thrones (season 1)*. Dec. 2023. URL: https://en.wikipedia.org/wiki/Game_of_Thrones_(season_1).

[218] Wikipedia. *Game of Thrones (season 2)*. Dec. 2023. URL: https://en.wikipedia.org/wiki/Game_of_Thrones_(season_2).

[219] Wikipedia. *Game of Thrones (season 3)*. Dec. 2023. URL: https://en.wikipedia.org/wiki/Game_of_Thrones_(season_3).

[220] Wikipedia. *Game of Thrones (season 4)*. Dec. 2023. URL: https://en.wikipedia.org/wiki/Game_of_Thrones_(season_4).

[221] Wikipedia. *Game of Thrones (season 5)*. Jan. 2024. URL: https://en.wikipedia.org/wiki/Game_of_Thrones_(season_5).

[222] S. Wiseman, A. M. Rush, and S. M. Shieber. "Learning Global Features for Coreference Resolution." In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Kevin Knight, Ani Nenkova, and Owen Rambow. 2016, pp. 994–1004. DOI: 10.18653/v1/N16-1114.

[223] T. Wolf et al. "Transformers: State-of-the-Art Natural Language Processing." In: *Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[224] A. Woloch. *The One vs. the Many: Minor Characters and the Space of the Protagonist in the Novel*. Princeton University Press, 2003. URL: https://press.princeton.edu/books/paperback/9780691113142/the-one-vs-the-many?srsltid=AfmBOopPzTHIz-18f5Dhh8kYMRJ6GMRU810bDzSPDc40DVakWlq1DI1_.

[225] Y. Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." In: *arXiv* cs.CL (2016), p. 1609.08144. URL: https://arxiv.org/abs/1609.08144.

[226] P. Xia, J. Sedoc, and B. Van Durme. "Incremental Neural Coreference Resolution in Constant Memory." In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 8617–8624. DOI: 10.18653/v1/2020.emnlp-main.695.

[227] K. Xie and M. Riedl. "Creating Suspenseful Stories: Iterative Planning with Large Language Models." In: *18th Conference of the European Chapter of the Association for Computational Linguistics*. Ed. by Yvette Graham and Matthew Purver. Vol. 1. 2024, pp. 2391–2407. URL: https://aclanthology.org/2024.eacl-long.147.

[228] L. Xu and J. D. Choi. "Adapted End-to-End Coreference Resolution System for Anaphoric Identities in Dialogues." In: *CODI-CRAC 2021 Shared Task on Anaphora, Bridging, and Discourse Deixis in Dialogue*. 2021, pp. 55–62. DOI: 10.18653/v1/2021.codi-sharedtask.6.

[229] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto. "LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention." In: *Conference on Empirical Methods in Natural Language Processing*. 2020, pp. 6442–6454. DOI: 10.18653/v1/2020.emnlp-main.523.

[230] H. Yan, T. Gui, J. Dai, Q. Guo, Z. Zhang, and X. Qiu. "A Unified Generative Framework for Various NER Subtasks." In: *59th Annual Meeting of the Association for Computational Linguistics and 11th International Joint Conference on Natural Language Processing*. Vol. 1. 2021, pp. 5808–5822. DOI: 10.18653/v1/2021.acl-long.451.

[231] L. Yang, L. Yuan, L. Cui, W. Gao, and Y. Zhang. "FactMix: Using a Few Labeled In-domain Examples to Generalize to Cross-domain Named Entity Recognition." In: *arXiv* cs.CL (2022), p. 2208.11464. DOI: 10.48550/ARXIV.2208.11464.

[232] Z. Yang et al. "State of the Art and Potentialities of Graph-level Learning." In: *arXiv* cs.LG (2023), p. 2301.05860. URL: https://arxiv.org/abs/2301.05860.

[233] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. "Tree of thoughts: deliberate problem solving with large language models." In: *37th International Conference on Neural Information Processing Systems*. 2024. URL: https://dl.acm.org/doi/abs/10.5555/3666122.3666639.

[234] C. Y. Yeung and J. Lee. "Identifying Speakers and Listeners of Quoted Speech in Literary Works." In: *8th International Joint Conference on Natural Language Processing*. Vol. 2. 2017, pp. 325–329. URL: https://aclanthology.org/I17-2055.

[235] J. Yose, R. Kenna, M. Mac Carron, and P. Mac Carron. "Network Analysis of the Viking Age in Ireland as portrayed in Cogadh Gaedhel re Gallaibh." In: *Royal Society Open Science* 5 (2018), p. 171024. DOI: 10.1098/rsos.171024.

[236] J. Yose, R. Kenna, P. Mac Carron, T. Platini, and J. Tonra. "A Networks-Science Investigation into the Epic Poems of Ossian." In: *Advances in Complex Systems* 19 (2016), p. 1650008. DOI: 10.1142/S0219525916500089.

[237] M. Zaheer et al. "Big Bird: Transformers for Longer Sequences." In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 17283–17297. URL: https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

[238] U. Zaratiana, N. Tomeh, N. E. Khbir, P. Holat, and T. Charnois. "GraphER: A Structure-aware Text-to-Graph Model for Entity and Relation Extraction." In: *arXiv* cs.CL (2024), p. 2404.12491. URL: https://arxiv.org/abs/2404.12491.

[239] W. Zhang, S. Wiseman, and K. Stratos. "Seq2seq is All You Need for Coreference Resolution." In: *Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. 2023, pp. 11493–11504. DOI: 10.18653/v1/2023.emnlp-main.704.

[240] X. Zhang, Y. Jiang, X. Wang, X. Hu, Y. Sun, P. Xie, and M. Zhang. "Domain-Specific NER via Retrieving Correlated Samples." In: *29th International Conference on Computational Linguistics*. 2022, pp. 2398–2404. URL: https://aclanthology.org/2022.coling-1.211.

APPENDIX

A.1 CHAPTER 3

A.1.1 *Novelties vo.1.o: OWTO Correction Process*

Dataset Issues

We find the following issues in the original OWTO dataset [59] [1]:

ANNOTATION ERRORS: We find false negatives where a known character is not annotated as an entity mention (e.g. `"d'Artagnan"` is sometimes annotated as `O`), and false positives where some tokens are incorrectly labeled as an entity mention (e.g. `"Quai de Ferraille"` is annotated `O B-PER O` in *The Three Musketeers*).

ANNOTATION INCONSISTENCIES: We find inconsistencies across the dataset novels. One prominent example is titles: depending on the novel, they are considered as part of the entity mention they are attached to or not (e.g. `"Mr. Frodo"` is annotated as either `O B-PER` or `B-PER I-PER`).

ENCODING ISSUES Some characters, such as accented ones, are incorrectly represented (e.g. `d'Artagnan` is sometimes represented as `dâ€™Artagnan`).

TOKENIZATION ISSUES There are a few tokenization issues. In the fantasy novel *The Wheel of Time*, characters with apostrophed names are tokenized in an unique way, where an apostrophe is considered a full token (e.g. `"Rand al'Thor"` would be tokenized as `[Rand, al, ', Thor]`). This rule is inconsistent with the rest of the dataset, and contrasts with the popular CoNLL-2003 dataset [195].

QUOTING ISSUES Quoting is handled inconsistently:

- Most books use double quotes to represent character dialogues and simple quotes for other purposes (irony, written text...), but some use the reverse convention.
- Quote pairs are usually started by backquotes (`` ` `` or `` ` ``) and end with simple quotes (`''` or `'`). This has the advantage that recursive quoting (i.e. quotes inside of quotes) can be correctly identified. However, the convention is sometimes incorrectly applied: some quotes start with simple quotes, and some end with backquotes.
- Some starting quotes are not matched with an ending quote. This can happen when the original book's presentation prevents any ambiguity in knowing the end of the quote (for example, when the quote ends a paragraph). However, the ambiguity is present in the NER dataset since the book presentation is not preserved by the CoNLL format.

1 We use the BIO tagging scheme for our examples, while the original dataset use the IO scheme.

Correction Methods

To fix annotation errors and inconsistencies in the annotation of PER mentions, and annotate LOC and ORG mentions, we adopt the following guidelines[2]:

- Titles preceding names are made part of entity mentions (Mr Bennet and Lord Eddard is annotated as B-PER I-PER)
- Names referring to families (Proudfoots or Bolgers in *The Fellowship of the Ring*) are annotated as persons.
- Ethnonyms and demonyms (such as Chyurda in *Assassin's Apprentice*) are annotated as O.
- Nicknames made evident by typography (for examples with capitalisation or dashes, such as You-Know-Who in *Harry Potter*) are annotated as entity mentions.
- Discontinuous mentions are annotated discontinuously (e.g. Mr and Mrs Bennet is annotated B-PER O B-PER I-PER).
- Capitalized common names (such as the Director in Brave New World, the Mouse in *Alice in Wonderland* or the Messenger in *The Painted Man*) are annotated according to the context. If the name is referring to a *function*, we annotate it as O. Otherwise, we treat it as a person.
- Nicknames following characters names are annotated as part of the entity mention (Gandalf the Grey are annotated B-PER I-PER I-PER). A notable exception is when the character is designated by including an organization or a location: in those cases, organizations and locations are annotated separately (Lord Stark of Winterfell is annotated B-PER I-PER O B-LOC, and Lord Eddard of House Stark is annotated B-PER I-PER O B-ORG I-ORG).
- Geopolical entity mentions that are also locations are annotated as ORG depending on context (France declared war over Germany is annotated B-ORG O O O B-ORG since only geopolitical entities can take action).

To correct existing PER annotations, we follow a semi-automatic error correction process, consisting of several steps applied in order:

FALSE NEGATIVES RULE We noticed a lot of false negatives in the existing annotation (e.g. Sherlock Holmes would be annotated O O). To fix those issues, we retrieve a list of characters names for all books. For each of those names, we also generate a list of alternative names using simple rules (e.g. {"Mr Sherlock Holmes"} => {"Mr Sherlock Holmes", "Mr Sherlock", "Mr Holmes", "Holmes", "Sherlock"}). Starting with longer names, when a mention of one of those names is not annotated as a person, we fix the annotation. Due to its high observed precision, we perform this step automatically, with no action from the annotator.

FALSE POSITIVES RULES We ask the annotator to correct the following cases if needed:

- When a series of tokens is annotated as a person, but is not found in the list of characters previously established.
- When a series of tokens is annotated as a person, but all of those tokens are lowercase.

2 These guidelines are *significantly* extended in the next version of Novelties, Novelties v1.0.0 [10].

BERT ASSISTED CORRECTION To catch the remaining errors, we finetune BERT [60] on a modified version of the CoNLL-2003 dataset [3]. We then ask the annotator to correct the annotation of spans of text if the prediction from BERT is different from the annotation.

MANUAL CORRECTION In last resort, we manually fix the remaining errors we could find.

QUOTING ISSUES To fix quoting issues, we apply the following guidelines :

- Character utterances should be enclosed in double quotes, starting with `` and ending with ''.
- Other use of quoting (such as, but not limited to: irony, written text, nicknames...) should be enclosed in single quotes, starting with ` and ending with '.
- A closing quote should be added when an utterance is not closed.

We once again adopt a semi-automatic method :

1. We execute a program that prompts the user for each single quote (since in some books, single quotes are used in place of double quotes). For each single quote, the user is able to choose if that quote should be replaced or not by an appropriate double quote, and if this choice for the current pattern (composed of the previous and the current token) should be automatically applied when seeing the pattern in the future.
2. We inspect quotes with less than 5 characters, or with more than 100 characters. Short quotes are often not part of dialogues, and we observe that a lot of them are enclosed in double rather than simple quotes. We inspect long quotes since there is a high chance these spuriously appear when an ending quote is missing.
3. We inspect quotes that have an uneven number of quotation marks, since these are malformatted.
4. We perform final corrections manually.

REMAINING ISSUES We fix the remaining tokenization and encoding issues manually. We also convert the dataset from the IO NER tagging scheme to the BIO scheme.

A.1.2 *Large Language Models Network Extraction Prompts*

Below, we present the prompt we use in Section 3.4.4 to extract character networks using 3 different LLMs: Llama3 [197], GPT3.5T [46] and GPT4o. We respectively present our prompts for the *LLM-Coref* and *LLM-E2E* extraction techniques.

3 Modifications consists of the inclusions of titles such as `Mr` or `Lord` as part of entity mentions, to remain consistent with our annotation guidelines

SYSTEM PROMPT You are an expert in literature and natural language processing.
USER PROMPT Given a text, you must extract characters and their mentions. Your answer must be the original text, where character mentions are tagged with the following format: [CHARACTER_ID]CHARACTER MENTION[/CHARACTER_ID]. You must tag character mentions only.

Here are some examples of this task:
Example 1:
Input: Elric was riding his horse . Alongside Moonglum , the prince of ruins was looking for his dark sword .
Output: [0] Elric [/0] was riding [0] his [/0] horse . Alongside [1] Moonglum [/1] , the [0] prince of ruins [/0] was looking for [0] his [/0] dark sword .

Example 2:
Input: Princess Liana felt sad , because Zarth Arn was gone . The princess decided she should sleep .
Output: [0] Princess Liana [/0] felt sad , because [1] Zarth Arn [/1] was gone . [0] The princess [/0] decided [0] she [/0] should sleep .

SYSTEM PROMPT You are an expert in literature and natural language processing.
USER PROMPT Given a text, you must extract a co-occurrence character network where vertices represent characters and edges represent their relationships. Each edge must have a weight corresponding to the number of interactions between two characters. Two characters without any interactions do not share an edge. An interaction between two characters occurs when two of their mentions occur within a distance of 32 tokens.
Your answer must be in a simplified Graphml-like format. Vertices must have an 'alias' attribute with the list of aliases of a character, separated by semicolons.

Here are some examples of this task:
Example 1:
Input: Elric was riding his horse . Alongside Moonglum , the prince of ruins was looking for his dark sword .
Output:

```
<graph>
<node id="n0"
 aliases="Elric;prince of ruins">
</node>
<node id="n1" aliases="Moonglum">
</node>
<edge id="e0" source="n0"
 target="n1"
 weight="2">
</edge>
</graph>
```

Example 2:
Input: Princess Liana felt sad , because Zarth Arn was gone . Liana decided she should sleep .
Output:

```
<graph>
<node id="n0"
 aliases="Princess Liana;Liana">
</node>
<node id="n1" aliases="Zarth Arn">
</node>
<edge id="e0" source="n0" target="n1"
 weight="2">
</node>
</graph>
```

A.2 CHAPTER 6

A.2.1 *Structural Matching*

Table 20 shows the results of structural alignment for all surveyed configurations. By comparison, Table 15 in Section 6.2 only focuses on the best results.

We can conclude that the Smith–Waterman algorithm performs better than thresholding. In addition, the best results are obtained with the `common` character set, whereas `top-20` per-

Table 20.: Narrative matching performance, expressed in F1-score, for all configurations of *structure*-based matching. The first column indicates the type of narrative unit used: text-constrained (*Text-Constr.*) or commensurate (*Commens.*). The *Repr.* column indicates whether the similarity is computed over vertex or edge sets. The *Measure* column indicates whether the similarity measure ignores weights (*Jaccard*) or take them into account (*Ružička*)

Narrative Units	Repr.	Measure	Alignment	*Novels* vs. *Comics*			*Novels* vs. *TV Show*			*Comics* vs. *TV Show*		
				named	common	top-20	named	common	top-20	named	common	top-20
Text-Constr.	Edges	Jaccard	Thresholding	25.86	28.46	11.66	12.86	13.18	6.59	39.85	44.08	31.28
			Smith–Waterman	54.55	53.85	38.49	28.38	**32.63**	1.54	47.78	46.93	45.71
		Ružička	Thresholding	29.61	34.74	13.31	20.72	23.43	7.34	43.36	46.95	33.33
			Smith–Waterman	55.63	57.54	43.84	25.43	24.12	1.75	39.55	49.72	46.86
	Vertices	Jaccard	Thresholding	13.73	17.61	8.53	11.67	13.72	6.15	28.30	24.31	19.85
			Smith–Waterman	53.71	56.54	26.71	28.72	23.61	8.04	39.77	49.16	39.33
		Ružička	Thresholding	15.75	26.57	6.98	13.38	18.73	6.81	35.20	34.74	19.19
			Smith–Waterman	58.74	**62.94**	46.81	25.91	23.61	9.87	49.72	**51.40**	38.42
Commens.	Edges	Jaccard	Thresholding	24.67	27.95	11.33	17.97	16.91	6.71	37.72	37.86	34.18
			Smith–Waterman	71.52	71.14	49.35	33.17	31.58	22.54	47.62	47.87	46.99
		Ružička	Thresholding	25.66	33.23	14.04	21.02	18.72	7.74	44.34	44.79	38.18
			Smith–Waterman	71.38	71.57	59.93	31.33	34.38	24.33	47.62	49.46	46.99
	Vertices	Jaccard	Thresholding	16.79	19.84	8.24	15.67	12.54	6.42	8.39	5.97	7.50
			Smith–Waterman	71.81	70.23	60.93	31.78	32.16	30.69	**52.91**	**52.91**	44.32
		Ružička	Thresholding	14.80	23.82	7.77	15.62	17.63	6.55	35.35	26.92	7.14
			Smith–Waterman	70.95	**72.30**	63.33	34.46	**35.09**	25.91	51.06	51.06	40.88

forms the worst. However, it is difficult to reach a conclusion for the other two parameters: compared objects (edges vs. vertices), and weight usage (Jaccard vs. Ružička).

A.2.2 *Hybrid Matching*

Table 21 shows all the results of narrative matching using the *hybrid*-based similarity for the *text-constrained* and *commensurate* narrative units respectively. When using the *commensurate* units, hybrid similarity improves the results compared to structural similarity by between 8.1 and 11.3 F1 depending on the media pair. Overall, we obtain the best overall matching performance using the *commensurate* units to perform hybrid matching.

A.2.3 *Novels vs. TV Show over* `U2`

In this section, we present the results of narrative matching for the *Novels vs. TV Show* media pair over the `U2` time period. By comparison, Section 6.2 focuses on `U5`. Since the TV show diverges more and more from the novels across seasons, we expect that aligning adaptations over the `U2` period is easier than over the `U5` period.

The best results over all configurations can be found in Table 22, while the best alignment can be found in Figure 31. We restrict ourselves to the best results per similarity due to the great number of possible configurations. As we expect, the performance is higher for the `U2` than the `U5` period, due to the divergence of the later seasons. We also observe that, as when aligning over `U5`, matching using commensurate units yields a positive performance boost, with a large increase of 17.9 F1. Combining textual and structural matching increases performance (+5.8 F1), but less than taking commensurate into account.

Table 21.: Narrative matching performance, expressed in F1-score, for all configurations of *hybrid*-based matching. The first column indicates the type of narrative unit used: text-constrained (*Text-Constr.*) or commensurate (*Commens.*). The *Text Sim.* column indicates the textual similarity measure in usage. The *Struct Repr.* column indicates whether the similarity is computed over vertex or edge sets. The *Struct Measure* column indicates whether the similarity measure ignores weights (*Jaccard*) or take them into account (*Ružička*)

Narrative Unit	Text Sim.	Struct. Repr.	Struct. Measure	Alignment	*Novels* vs. *Comics*			*Novels* vs. *TV Show*			*Comics* vs. *TV Show*		
					named	common	top-20	named	common	top-20	named	common	top-20
Text-Constr.	*sbert*	Edges	Jaccard	Thresholding	27.00	18.60	6.90	17.33	15.69	7.04	40.13	42.70	24.90
				Smith–Waterman	63.16	63.86	61.75	27.06	27.41	1.53	38.86	45.71	44.07
			Ružička	Thresholding	11.54	15.95	13.79	22.58	24.64	10.98	45.09	46.75	25.17
				Smith–Waterman	57.75	60.35	58.95	25.16	26.21	17.80	39.55	40.68	38.42
		Vertices	Jaccard	Thresholding	25.64	9.09	13.23	11.43	12.00	6.32	32.22	33.99	22.22
				Smith–Waterman	65.25	62.90	61.97	22.11	28.01	12.50	35.29	35.29	32.94
			Ružička	Thresholding	29.32	29.51	13.78	15.79	20.07	6.32	36.67	46.15	22.73
				Smith–Waterman	62.46	65.96	57.45	28.95	27.40	21.73	44.07	44.71	32.94
	tfidf	Edges	Jaccard	Thresholding	27.51	20.93	7.73	19.95	22.57	12.65	34.10	35.40	25.42
				Smith–Waterman	57.24	57.54	58.95	30.24	29.31	9.89	47.78	42.46	42.94
			Ružička	Thresholding	21.18	33.67	14.96	23.73	26.90	14.02	33.72	39.14	24.97
				Smith–Waterman	60.28	60.28	58.95	29.10	27.09	17.02	46.93	42.46	42.94
		Vertices	Jaccard	Thresholding	24.31	19.51	14.69	15.06	13.00	12.80	31.00	29.90	24.73
				Smith–Waterman	61.70	60.99	51.06	21.93	26.95	16.62	42.46	48.04	31.14
			Ružička	Thresholding	38.74	39.60	15.17	20.50	22.60	12.31	33.59	39.29	26.89
				Smith–Waterman	65.26	**67.37**	47.52	30.61	**30.95**	16.44	**49.16**	**49.16**	31.14
Commens.	*sbert*	Edges	Jaccard	Thresholding	29.41	39.06	8.28	18.08	21.40	6.82	44.28	46.24	25.85
				Smith–Waterman	70.71	71.19	70.51	33.50	34.26	30.20	53.68	58.64	**63.87**
			Ružička	Thresholding	43.27	45.98	13.26	23.58	22.58	9.87	39.41	43.81	27.05
				Smith–Waterman	72.11	70.75	66.22	36.86	36.43	30.50	58.33	52.08	60.73
		Vertices	Jaccard	Thresholding	7.84	11.25	13.14	17.30	14.96	6.13	37.35	33.72	25.23
				Smith–Waterman	69.39	75.93	68.90	32.36	35.86	31.78	59.38	60.42	61.46
			Ružička	Thresholding	21.05	47.62	12.58	16.49	19.17	6.25	32.36	33.99	25.79
				Smith–Waterman	72.60	74.75	70.43	34.04	33.63	35.56	59.07	54.74	61.46
	tfidf	Edges	Jaccard	Thresholding	22.10	31.96	13.26	19.17	15.62	11.71	31.96	31.58	24.56
				Smith–Waterman	72.97	74.07	75.84	32.58	35.79	33.29	59.69	58.33	52.08
			Ružička	Thresholding	27.32	45.91	12.57	22.39	21.37	12.46	32.34	33.28	24.30
				Smith–Waterman	**78.50**	74.50	74.92	35.00	38.30	33.62	58.03	58.03	56.54
		Vertices	Jaccard	Thresholding	17.05	20.77	12.22	20.69	15.25	12.75	32.38	28.07	22.99
				Smith–Waterman	71.86	74.92	70.00	34.30	36.27	29.97	55.96	59.69	46.07
			Ružička	Thresholding	35.24	46.84	13.33	16.91	21.08	11.92	29.58	31.18	23.37
				Smith–Waterman	75.00	74.75	72.48	36.18	**39.04**	31.27	55.96	58.33	46.07

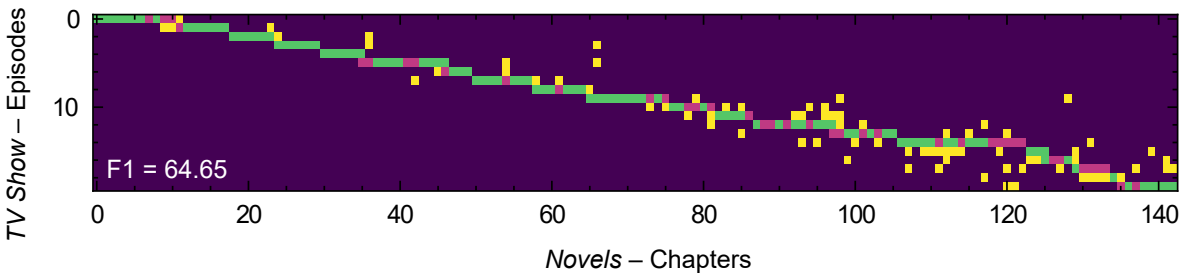


Figure 31.: Best performing alignment for the *Novels vs. TV Show* pair over time period U2.

Table 22.: Performance obtained on the U2 time period of the *Novels vs. TV Show* pair, expressed in terms of F1-score. Only the best results across configurations are shown.

Alignment	Textual Matching	Structural Matching		Hybrid Matching	
		Text-Constrained	Commensurate	Text-Constrained	Commensurate
Thresholding	29.01	37.24	32.56	37.95	41.18
Smith–Waterman	36.65	45.00	62.87	50.78	**64.65**

A.2.4 *Cumulative Networks*

Table 23 presents results of structural narrative matching using *cumulative* networks on the text-constrained narrative units. By comparison, Section 6.2 focuses on *instant* networks.

Table 23.: Performance obtained when using *structural*-based representations using *cumulative networks* and the *text-constrained* narrative units to tackle the narrative matching task, expressed in terms of F1-score. Columns are organized as in previous tables.

Representation	Measure	Alignment	*Novels* vs. *Comics*			*Novels* vs. *TV Show*			*Comics* vs. *TV Show*		
			named	common	top-20	named	common	top-20	named	common	top-20
Edges	Jaccard	Thresholding	4.05	3.94	3.61	6.36	5.12	4.89	19.98	20.20	19.95
		Smith–Waterman	16.85	41.40	10.79	2.00	22.34	0.00	19.88	29.71	21.05
	Ružička	Thresholding	3.80	3.98	3.62	5.66	5.17	4.90	20.21	20.41	20.22
		Smith–Waterman	22.14	**47.18**	11.72	7.75	13.11	1.27	18.29	26.59	21.05
Vertices	Jaccard	Thresholding	4.10	3.90	3.53	4.82	5.18	4.89	22.24	20.78	19.79
		Smith–Waterman	12.69	40.43	1.01	1.19	**25.63**	0.42	20.25	33.33	7.74
	Ružička	Thresholding	3.76	3.88	3.53	7.72	5.19	4.89	20.81	20.23	19.95
		Smith–Waterman	17.02	32.62	1.01	2.07	24.97	0.42	22.50	**35.63**	7.74

Since the cumulative network of a narrative unit is the aggregation of previous instant networks, differences between narrative units are less and less noticeable, leading to poor matching performance. The Smith–Waterman algorithm is still able to achieve mild performance in some cases (47.18 F1 on the *Novels vs. Comics* pair), but overall matching with instant networks clearly yields more performance.