FRI Natural language processing course 2021
Project proposal

University of Ljubljana
Faculty of Computer and Information Science

Literacy situation models knowledge base creation

Jan Krivec, Matija Gerčer and Rok Bosil

Abstract

This paper presents our implementation, results and discussion of a model, which finds the main characters for a given book and defines, if they are allies or enemies. In the introduction we first present our problem and list similar articles in related work. Next we describe the data we used and give some basic statistics. In chapter Methodology we describe how we defined and built the model, and describe method used in each step. In the end we write our results and conclusion.

Keywords
Natural language processing, Knowledge base

Advisors: Slavko Žitnik

Introduction

The field of natural language processing largely focusses on approaches of processing, analyzing and understanding large amounts of text. Namely, we can extract important information from texts that can have a great added value for both science and the economy. In this project, we will focus on models for building a knowledge base from books. More precisely, we will create a model that tries to find all the characters in a book and then determine the alliances, i.e. which characters are allies and which are enemies. For our dataset we chose first three novels of a well known series A Song of Ice and Fire: A Game of Thrones, A Clash of Kings and A Storm of Swords. In our experimentation we will focus more on the first novel, where we will split it into chapters and then visualize the development of character relationships throughout the story. For our model to work, we first need to preprocess the data to prepare it for further analysis. Our model then consists of two parts. First we find all named entities and filter them, so we are left only with characters. In the second part we use sentiment analysis to determine if characters mentioned together are allies or enemies. For each task we tried using several different algorithms and compared them between eachother.

Related work

In [2] the authors focused on extraction of information about literacy characters. One of their tasks was to indentify and classify the main characters in folktales. They used Propps model, to identify main characters and their roles in folktales. In their description of used ontology, they defined three knowledge sources: *Folktale morphology*, which relies on the Propp's model. This splits characters into nine types with the use of axioms. Next *folktale main entities* and axioms for their recognition were formalized. The axioms depict the animals, witches and supernatural characters. Last, they had to define *family relationships in folktale*. The ontology was processed using OWL API to generate classes of characters from the ontology into GATE. With the help of part of speech (POS) anotation labels and named entity recognition (NER) using conditional random fields (CRF) sequence taggers to retrieve the characters. The authors used three sequential algorithms for retrieving knowledge about characters. Main goal of these algorithms was to: extrach characters from folktales, use decoreference resolution and in the end finding the perspective of characters.

In [4] the authors propose a hybrid method of main character indetification and extraction of relationships between them. This approach mainly combines supervised and unsupervised learning methods. In contrary to unsuperised learning, supervised learning algorithms learn to classify with the help of already known realtion types between entities.

Data

We focused our analysis on the first three books of A Song of Ice and Fire book series. The series is a series of epic fantasy novels written by George R. R. Martin. There are seven books planned for the whole series, but as of the time of writing this report only five have been finished. In our report we will focus on the analysis of the first three books of the series; A Game of Thrones (AGOT), A Clash of Kings (ACOK) and A Storm of Swords (ASOS).

Book analysis

We analysed all the books, but more focus was given to the fist book. Here we state some important and interesting information about A Game of Thrones.

The A Game of Thrones book is composed of 74 chapters. All the chapters but two, the first which is a prolog chapter and the last which is an appendix, are written in a point of view (POV) of a certain character. There are eight POV character in the first book; **Bran**, **Catelyn**, **Daenerys**, **Eddard**, **Jon**, **Arya**, **Tyrion** and **Sansa**.

For a more in depht analysis we split the book into chapters.

Methodology

Our pipeline consists of three parts:

1. Preprocessing
2. Entity extraction
3. Sentiment analysis between entities

Preprocessing

We decided that we will perform analysis on all three books, but will focus more on the first book titled A Game of Thrones. As metioned before we split the book into 74 chapters. The story input is preprocessed before passing it into different phases. We used basic preprocessing, where we removed all special characters. We did not convert text to lower case, as that provided worse results for named entity recognition.

Coreference resolution

Before we can extract entities from text, we need to perform coreference resolution. Coreference resolution is the task of finding all referring expressions (he, she, I, that, this..., or any subject or noun) that refer to the same real-world entity. For coreference resolution we used the spaCy and Allennlp libraries.

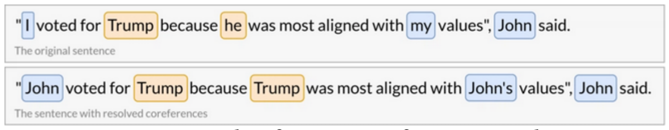

"I voted for Trump because he was most aligned with my values", John said.
The original sentence
"John voted for Trump because Trump was most aligned with John's values", John said.
The sentence with resolved coreferences

Figure 1. Example of using coreference resolution

Replacement of synonyms

With the help of A Wiki of Ice and Fire [1] we made a dictionary of synonyms for each important character in the book. With its help we replaced all the synonyms of a certain character with chosen general name. This helped with the count of the actual number of occurances of a certain character in the book. We also tackled the problem of characters with the same first name. For example: Jon Snow and Jon Arryn, we replaced all Jon Arryns with a specific string so that we will not wrongly detect named entities.

Entity extraction

For finding characters we used Named entity recognition (NER). NER is a type of text processing, where our main goal is to locate and classify named entities into predefined categories. These include $PERSON$, $ORGANIZATION$, $LOCATION$ and more. In our project we used many NERs from different models/libraries, such as Flair, Nltk, Spacy, Stanza and more. Out of all named entities, we focused on recognising people for retrieving our character list.

Ner - Flair

Flair is a powerful NLP library, that allows the application of state-of-the-art natural language processing models on text, such as NER, part-of-speech tagging (PoS), special support for biomedical data, sense disambiguation and classification, with support for a rapidly growing number of languages. Flair can also be used as a text embedding library, which means it is used for representing text (own embeddings, as well as BERT embeddings and ELMo embeddings). It is also built directly on PyTorch, which makes it easy to train own models [14]. Flair representations are a bi-LTSM character based monolingual model pretrained on Wikipedia.

Ner - Spacy

Spacy is an open-source library for advanced natural language processing in Python. It's designed more for production use rather than research use and it's very user-friendly. Despite that contains a lot of different models and supports many foreign languages.

Ner - Nltk

Natural Language Toolkit (NLTK) is a suite of libraries and programs for statistical and symbolic natural language processing for the English language. NLTK can be used for different language processing tasks, such as tokenization, parse tree visualization, NER and more. We used NLTK in data preprocessing and named entity recognition.

Ner - Stanford

Stanford NER is a Java implementation of a NER that can be run via a Python interface. [16] In this article, we are using version 4.2.0. Unfortunately, the NER can't distinguish between first and last name and is therefore difficult to compare with the other NER models.

Ner - Stanza

Stanza is a collection of accurate and efficient tools for linguistic analysis which also contains advanced NER models (8 languages). The modules are built on top of the PyTorch library and its components also enable efficient training and evaluation with custom annotated data. [15]

Sentiment analysis

After named entity recognition the whole book is preprocessed and segmented at sentence level.Sentences are selected if they include the specified tuple of characters in it. Selected sentences are then passed to be evaluated with sentiment analysis.

Sentiment analysis is used to identify and extract subjective information and emotional tone behind a body of text. We used several different libraries for sentiment analysis.

Afinn

Afinn is a simple lexicon, that contains more than 3000 words with a polarity score associated with each word. Sentence is scored as a combined score of all words in the sentence.

TextBlob

TextBlob is another lexicon-based approach, where sentiment is defined by its semantic orientation and the intensity of each word in the sentence, which requires a predefined dictionary of positive and negative words. TextBlob returns polarity and subjectivity of a sentence, where subjectivity presents amount of personal oppinion contained in text.

Vader

Vader (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that has pre-defined rules for words or lexicons. With the use of these rules Vader can detect negative sentiment better than TextBlob, because it can detect otherwise hidden negative sentiment better.

Sentiment analysis models have different score boundries (Afinn -6.0 – 6.0, Textblob -1.0 – 1.0 and Vader -4.0 – 4.0), but in every model negative scores represent negative sentiment and positive scores represent positive sentiment. We can see some examples in table 1.

Sentence	Afinn	Textblob	Vader
Today is a wonderful day.	4.0	1.0	0.57
The food was great, but I didn't like the service.	5.0	0.8	-0.03
I didn't really enjoy the music.	2.0	0.4	-0.43

Table 1. Comparison of sentiment scores using different sentiment analysis libraries.

Results

Coreference resolution

We tried both spaCy and Allennlp libraries for coreference resolution. Bellow we can see one example of the use of both libraries.

Normal text: *"We should start back," Gared urged as the woods began to grow dark around them. "The wildlings are dead." "Do the dead frighten you?" Ser Waymar Royce asked with just the hint of a smile. Gared did not rise to the bait. "Dead is dead," he said. "Are they dead?" Royce asked softly.*

Spacy: *"We should start back," Gared urged as the woods began to grow dark around Gared. "The wildlings are dead." "Do the dead frighten you?" Ser Waymar Royce asked with just the hint of a smile. Gared did not rise to the bait. "Dead is dead," Gared said. We have no business with the dead. Are they dead? Ser Waymar Royce asked softly.*

Allennlp: *"We should start back," Gared urged as the woods began to grow dark around We. "The wildlings are dead." "Do the dead frighten Gared?" Ser Waymar Royce asked with just the hint of a smile. Gared did not rise to the bait. "Dead is dead," Gared said. "Are The wildlings dead?" Ser Waymar Royce asked softly.*

As we can see from the results both models were able to replace some of the pronouns. Both were able to replace the abreviation "Royce" with "Ser Waymar Royce" in the last sentence. Allennlp performed better though, because it was able to find "you" and "they" and replace them with correct entities. This is why we used Allennlp in our further analysis.

Named entity recognition

In this part, we will present five different NER models that we have implemented and evaluated in different ways. We tested the models on our corpus data as well as on NLP's well-known Cornell 2003 dataset.

Unique names - GoT - Chapter 1

The first part of the NER analysis focuses on finding the characters in the book. For this task, we have built several different models, which we ran on the first chapter of the book. The top five results according to the number of occurrences in the text are presented in the following table. The table shows that the Flair and Stanza models performed best, followed by the Stanford and Nltk NER models. The worst model here was Spacy, which didn't detect many characters at all. For example - throughout the first chapter, Flair detected 28 unique characters, a total of 162 entities, meanwhile Spacy detected only 17 unique characters with a total of 45 entities. Also, we would like to point out that the Spacy NER detected Bran only once while Flair and Stanza as many as 46 times

FLAIR	SPACY	NLTK	STANFORD	STANZA
Bran (46)	Jon (15)	Bran (42)	Jon (31)	Bran (46)
Robb (29)	Greyjoy (4)	Jon (29)	Robb (29)	Robb (29)
Jon (28)	Hullen (4)	Robb (27)	Greyjoy (12)	Jon (29)
Jory (9)	Stark (3)	Father (8)	Jory (11)	Jory (9)
Greyjoy (7)	Jon Snow (3)	Jory (7)	Theon (7)	Greyjoy (7)

Table 2. Top five NER models results according of to the number of occurrences - Chapter 1

NER evaluation - GoT - Chapter 1

In the next part of the NER analysis, we focused on a more formal evaluation approach of the implemented models. For the first chapter of the book, we annotated sentences manually and took them as a golden reference. Character names were annotated with two entity types - depending on whether they appeared at the beginning or at the end of the character's full name (e.g. Eddard Stark -> Eddard [B-PER] Stark [I-PER]). After that, we compared the actual annotated data with data returned by our NER models. We observed how many times the model predicts correctly and how many times they make a mistake. More precisely, we observed the next sentence situations:

- Correct (COR) - Both words are the same
- Incorrect (INC) - The output of a model and the golden annotation didn't match
- Partial (PAR) - Model and the golden annotation were somewhat "similar" but not the same
- Missing (MIS) - A golden annotation was not captured by a model
- Spurius (SPU) - Model produced a response which didn't exit in the golden annotation

According to the numbers in the table, we can see that the Flair model worked the best due to a small number of missing and spurious errors. Stanza and Nltk models also performed well with slightly less corrected predictions while the Spacy model returned poor results. From the table, we can also see Precision (P), Recall (R) and F1 scores, which were calculated via actual - ACT (TP + FN) and possible - POS (TP + FP) values.

	FLAIR	**SPACY**	**NLTK**	**STANFORD**	**STANZA**
COR	156	44	137	102	149
INC	0	0	0	0	0
PAR	0	0	0	0	0
MIS	8	120	27	62	15
SPU	7	1	7	27	13
POS	164	164	164	164	164
ACT	163	45	144	129	162
P	0.9571	0.9778	0.9514	0.7907	0.9197
R	0.9512	0.2683	0.8354	0.6220	0.9085
F1	0.9541	0.4211	0.8896	0.6962	0.9141

Table 3. Different metrices of different NER models - Chapter 1

NER evaluation - Conll2003

In the last part of the NER analysis, we run the evaluation method on a different dataset - Conll 2003, which is slightly more challenging for NER and reflects a more realistic assessment of performance. The results are presented in a similar table as before, with slightly different. Because the Stanford model doesn't separate entities into 2 groups, the model is not directly comparable with other NER models. Despite that, we can see that the Flair model again performed the best.

	FLAIR	**SPACY**	**NLTK**	**STANFORD**	**STANZA**
COR	1746	1103	1415	2644	1575
INC	0	0	0	0	0
PAR	0	0	0	0	0
MIS	96	739	427	493	267
SPU	31	167	590	391	152
POS	1842	1842	1842	3137	1842
ACT	1777	1270	2005	3035	1727
P	0.9826	0.8685	0.7057	0.8712	0.9120
R	0.9479	0.5988	0.7682	0.8428	0.8550
F1	0.9649	0.7089	0.7356	0.8568	0.8826

Table 4. Different metrices of different NER models - Conll 2003

Main characters

Before we performed sentiment analysis we wanted to know the "main character" of each chapter of the first book. As mentioned before there are eight POV characters in A Game of Thrones. We wanted to detect the main character of each chapter and find out whether the POV character is indeed the main character of its chapter.

The main character is determined based on the number of his/hers occurances in the chapter. For every chapter we performed above mentioned preprocessing. We tested the results where we did not use coreference resolution (normal) and when we used coreference resolution (once spaCy and once Allennlp). In every test we also replaced the synonyms. Because Flair produced the best results out of all the Named entity recognition methods we only used that one. We detected all the named entities and ordered them based on the number of occurances.

Character	**Number of POV chapters**	**Average position**
Bran	7	1
Catelyn	11	1.36
Daenerys	10	1
Eddard	15	1.13
Jon	9	1
Arya	5	1
Tyrion	9	1
Sansa	6	1

Table 5. Main characters for each chapter of the A Game Of Thrones book and comparison to the POV character of the chapter. These are the results with Allennlp coreference resolution.

In Table 5 we can see the results of our testing with Allennlp coreference resolution. We can see that every POV character, except Catelyn and Eddard, is the main character for each of their POV chapters. In the case of Catelyn, she is not the main character in three chapters (once she is the third most common character and twice the second most common character) and in the case of Eddard; he is the third most common character in one of his POV chapters.

For every instance when the character is not the main character of the chapter it is very close to the most common character. (For example in one Catelyn chapter Eddard is first with 102 occurances and Catelyn is second with 98 occurences.)

We also looked at the results with spaCy coreference resolution and without coreference resolution. With both approaches we get very simmilar results. Without coreference resolution only Catelyn is not the main character in all of the POV chapters (she is second three times). With spaCy coreference resolution we get the worst results, but still almost all the characters are the main characters of their POV chapter, if not they are in the second or third position.

We also looked at the main character of the entire novel. Bellow are the top three characters in terms of occurances for the three different approaches (number of occurances is in brackets):

- **Normal:** Eddard (843), Jon (780), Tyrion (608), ...
- **spaCy:** Jon (1843), Arya (1643), Eddard (1548), ...
- **Allennlp:** Jon (1284), Tyrion (1249), Eddard (1213), ...

There are a lot of characters and also POV characters in the novel, but some are more prevelant than others. All the above top characters are very simmilar in terms of occurances in all three approaches.

Sentiment analysis

We divided sentiment analysis into two parts, co-occurrence and relationship between characters. First we had to decide which characters to compare with each other. We decided that we will take 21 most common characters in the whole novel and perform sentiment analysis on them. We calculated relationships and co-occurrence for chapters in batch of 8, so that we could see the porgression of relationships through the novel. For the 21 most common characters in the novel we calculated their number of occurrences for the specific batch of chapters and perfomed sentiment analysis.

For calculating sentiment and co-occurrence we first created a matrix of all occurrences of main characters with the help of CountVectorizer from *Scikit-learn* library, where we passed the names of main characters as vocabulary. To obtain the co-occurrence matrix, we then vectorized the computed matrix and performed dot product with its transpose. Co-occurrence graph can be seen in figures 2 and 3.

For computing the sentiment score, we multiplied the obtained co-occurrence matrix with the scores recieved from sentiment analysis models.

Because some authors might have a more negative tone of writing, or if the book is written with more negative emotion descriptors, we also introduced **Sentiment alignment rate**. It is used to align the sentiment scores. Everytime two same characters are mentioned in the same sentence, the sentiment score between them will change by a small degree. We defined this alignment rate as negative two times the mean sentiment score of non-zero-score sentences in the whole novel. This mean score defines if the book has a negative or positive tone. Before returning the sentiment matrix we sum it with dot product of co-occurrence with the allignment score. As this score is negative it will negate a certain tone in the book to some extent. We multiplied this rate with 2, as co-occurrence between characters does not happen in every sentence.

We tested our model on three Game of Thrones books. We used the networkx library to plot the graph. We used the sentiment matrix for the weights on the edges of the graph. We tried running our algorithm on the first book of A Song of Ice and Fire series: A Game of Thrones. The results can be seen in figures 4 and 5. Blue edges represent allies and red edges represent enemies. The stronger the color is, more that relationship is expressed. If link is not visible, or is barely visible, that means that the relationship between those two characters is neutral.

Gifs of the sentiment analysis can be found on our github page https://github.com/bosilr/NLP_project.

Discussion

Results received for coreference resolution and NER were good.

Obtained relations displayed in the knowledge graphs seem logical. After we use coreference resolution, graphs change as expected, but the relations normally get even more expressed or slightly change. There are very few negative relations expressed. That is probably due to the fact that bot Afinn library is small and based on tweets and the vocabulary used in novels is different and more complex.

There are a lot more negative relations expressed when using Vader for sentiment analysis. When analysing text, which was preprocessed using coreference resolution, there are even more negative relations. There are some illogi-

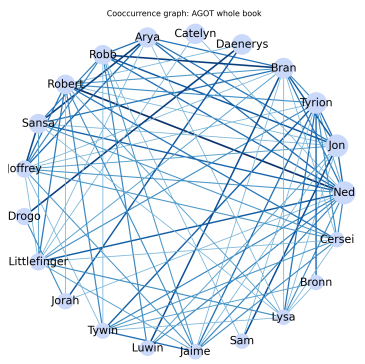


Figure 2. Co-occurrence graph.

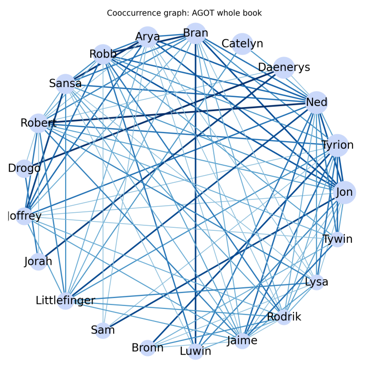


Figure 3. Co-occurrence graph using coreference resolution. We can see that there are more and stronger connection between characters than in figure 2.

cal connections visible for instance Bran and Luwin, which sould be good friends, or Jon and Ned Stark, but at closer inspection this can be reasoned with to an extent, as we found cases of Jon being called Ned's bastard son. Arya and Bran also should not be enemies. Otherwise most of the connections seem logical. We can see that in the same household characters are normally allies. For instance Jon and Robb Stark or Sam, or Arya and Sansa, where connection is neutral as they are sisters, but did not like eachother very much. There are also good relations between Joffrey and Sansa, which were going to marry.

We can say that overall our algorithm for detecting allies worked well, as the knowledge graph presented does make sense.

We could improve the relation extraction by defining our own relations and training a model better suited for this corpus, as sentiment can be deceptive in books with a more negative tone. We could also try to combinate current libraries for sentiment analysis.

Conclusion

In this paper we presented implementation and analysis of a model for knowledge base creation. More specifically a model for finding the main characters in a book and determening if thes are allies or foes. We found, correference resolution using Allennlp library was useful and provided good results. For character extraction using NER the best library proved to be Flair and for sentiment analysis libraries Afinn and Vader provided good results, both being better than another for some specific character combination. Relationship extraction could be improved by using an own trained model with specific relations. Next step could also be performing community detection on the obtained graphs.

References

[1] A wiki of ice and fire. https://awoiaf.westeros.org/index.php/Main_Page.

[2] Adrian Groza and Lidia Corde. Information retrieval in folktales using natural language processing, 2015.

[3] Sebastian Ruder. http://nlpprogress.com/english/relationship_extraction.html.

[4] V. Devisree and P.C. Reghu Raj. A hybrid approach to relationship extraction from stories. *Procedia Technology*, 24:1499–1506, 2016. International Conference on Emerging Trends in Engineering, Science and Technology (ICETEST - 2015).

[5] Natalia Konstantinova. Review of relation extraction methods: What is new out there? In *Analysis of Images, Social Networks and Texts*, pages 15–28, Cham, 2014. Springer International Publishing.

[6] Yifan Peng, Manabu Torii, Cathy H. Wu, and K. Vijay-Shanker. A generalizable nlp framework for fast development of pattern-based biomedical relation extraction systems. *BMC Bioinformatics*, 15(1):285, Aug 2014.

[7] Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st workshop on vector space modeling for natural language processing*, pages 39–48, 2015.

[8] Suncong Zheng, Yuexing Hao, Dongyuan Lu, Hongyun Bao, Jiaming Xu, Hongwei Hao, and Bo Xu. Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing*, 257:59–66, 2017. Machine Learning and Signal Processing for Big Multimedia Analysis.

[9] Jinhua Dou, Jingyan Qin, Zanxia Jin, and Zhuang Li. Knowledge graph based on domain ontology and natural language processing technology for chinese intangible cultural heritage. *Journal of Visual Languages Computing*, 48:19–28, 2018.

[10] Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu, and Huajun Chen. Label-free distant supervision for relation extraction via knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2246–2255, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[11] hzjken. Character network. https://github.com/hzjken/character-network.

[12] Rolf A. Zwaan. Situation models: The mental leap into imagined worlds. *Current Directions in Psychological Science*, 8(1):15–18, 1999.

[13] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Push-

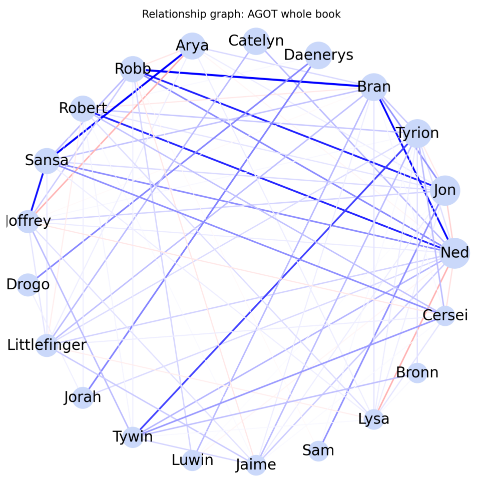
meet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics.

[14] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

[15] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.

[16] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Figure 4. Relation graph using Afinn library for sentiment analysis.



(a) Original text.

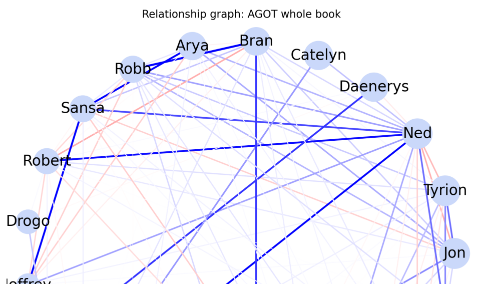
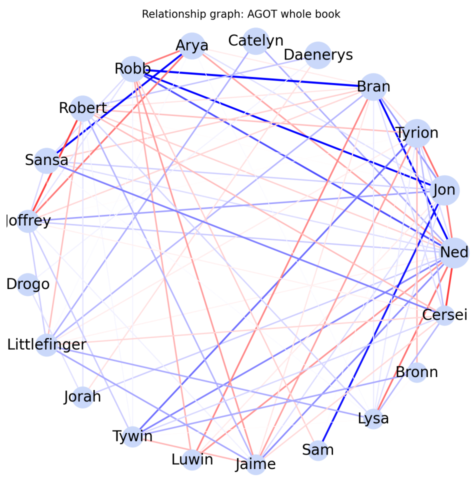
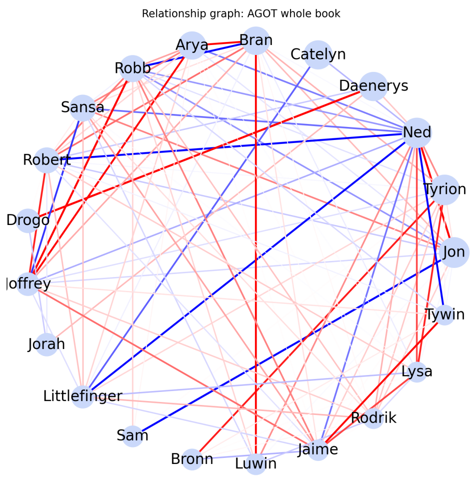


Figure 5. Relation graph using Vader library for sentiment analysis.



(a) Original text.



(b) Modified text with Allennlp coreference resolution.