FRI Natural language processing course 2023
Project report

University of Ljubljana
Faculty of Computer and Information Science

Literacy situation models knowledge base creation

Mila Marinković, Ilina Kirovska and Radoslav Atanasoski

Abstract

Short stories are powerful mediums that in spite of their length often convey complex themes and ideas. Understanding them demands a thorough grasp of the context in which they were written. The purpose of this project is to help the readers in bettering their understanding and knowledge of the stories by building a knowledge base. Using NLP techniques such as named-entity extraction and sentiment analysis we managed to identify the stories' character relationships and their sentiments as well as identify the protagonists and antagonists. Finally, we have made graphs to analyze the relations and sentiments of communicators from selected stories.

Keywords
character extraction, NER, coreference resolution, sentiment analysis, data visualization, short stories

Introduction

Analyzing literary works and their content is a complex task for machines, as they lack human knowledge, common sense, and contextual awareness crucial for such analysis. In this research, we focus on a subset of literary analysis, specifically fictional character analysis, which poses significant challenges. Our approach addresses the extraction of characters, sentiment analysis of character relationships, and the detection of protagonists and antagonists.

This paper aims to construct a knowledge base using situation models derived from selected English short stories. We propose a simple pipeline that encompasses named entity extraction, character sentiment analysis, and the exploration of character co-occurrence and relationships. By extracting fictional characters from short literature excerpts, conducting sentiment analysis on all characters, and studying the co-occurrence of recognized characters, our objective is to provide meaningful and accurate character analysis for short stories.

The ultimate goal of this research is to offer a comprehensive and insightful analysis of fictional characters in short stories. Through the application of our proposed pipeline and leveraging our annotated dataset, we strive to provide a valuable resource for character-centric literary analysis, enabling a deeper understanding of narrative dynamics and character interplay in short story contexts.

Related work

In the paper [1], the supervised and unsupervised approaches are integrated to produce a hybrid model that can extract relationships from stories. The model identifies the main characters, gathers sentences that are relevant to them, analyzes these words, and then assigns a classification to the characters' relationships. If a sentence is classified during training, it passes through the supervised section of the model; otherwise, it goes through the unsupervised section. The corpus utilized was a collection of 100 children's short stories that featured a variety of different relationship types. This model identified if a relationship between characters was "Parent-Child", "Friendship", or "No-Relationship". When compared to other approaches, the hybrid model performed well.

A rule-based approach to character identification and family connection analysis is presented in the paper [2]. The authors used fundamental NLP procedures such as tokenization, POS tagging, and sentence parsing to extract the entities and relationships among them. There are two key modules i.e. Character extraction and Relationship extraction. For the character extraction, the NER kit provided by NLTK was used to determine the possible characters. Ultimately, from all results, the entities labeled as 'persons' were identified as characters. Regarding the relationship extraction parse tree generation was used where each entity was given a few properties including relation id. Following the pronoun resolution and character extraction, the relation id of each entity is modified during the execution of relation resolution on each parse tree. After the identification of some relationships, new relationships are resolved using the propagation rules for relationships. Considering that character extraction accuracy equaled 95.33% and relationship extraction accuracy to 80.5% it is evident that this approach was effective.

Dataset

Since we were not able to find a dataset that was good enough for our needs, we decided on creating our own by manually annotating stories. We searched for appropriate stories on Project Gutenberg [3], which is a digital library containing a large number of eBooks. "Grimm's Fairy Tales" [4] is a collection of fairy tales and folklore stories, compiled by brothers by Jacob and Wilhelm Grimm. A fairy tale is a short story that falls under the folklore category. These tales typically involve magic, enchantments, and fictitious or mythical creatures. Besides their length making them easier to annotate the fact that we were already familiar with most of the stories in this book led us to the decision that they will be our main data.

For each story, we generated two files, one text file containing the raw text of the story and a JSON file that contains the annotations. The annotations can be split into three parts:

1. Characters - contains the story characters.
2. Relationships - contains the relationships between characters.
3. Protagonist and antagonist - the story's protagonist and antagonist, if there is one.

We read and evaluated each of the selected stories to create these annotations and we provided an example of one such file for a Cinderella fairy tale on Listing 1. Because of its length, we only showed a part of the file. The relationships between characters can be "negative", "neutral" or "positive". In the JSON file, these relationship types are represented by -1, 0, and 1 respectively.

Listing 1. Part of the annotation for "Cinderella".

```
{
    "Characters": ["cinderella", "stepmother
       ", "sisters", "godmother", "prince",
       "king", "queen"],
    "Relationships": {
        "cinderella": [
            ["stepmother", -1],
            ["sisters", -1],
            ["godmother", 1],
            ["prince", 1]
        ],
        "stepmother": [
            ["cinderella", -1],
            ["sisters", 1]
        ]
    },
    "Protagonist": "cinderella",
    "Antagonist": "stepmother"
}
```

From the given example of an annotation file of the Cinderella story, we can see who are the story characters in the field "Characters". Those characters were later taken as ground-truth characters. Next, the field "Relationship" represents the evaluation of the sentiment between one character and the other characters that he is related to, marked with the sentiment -1, 0, or 1, depending on their relationship. Finally, the last two fields "Protagonist" and "Antagonist" obtain the name of the character that is protagonist viz. antagonist.

For the purpose of this project, we collected a dozen stories, while their annotations can be seen in this project git repository.

Methodology

The main idea of our project is to extract characters from the previously mentioned data, build a character network on top of that and detect the protagonists and antagonists. The project pipeline consists of the following stages:

1. Character extraction
2. Coreference resolution (CR)
3. Sentiment analysis
4. Visualization

Character extraction

Character extraction represents a baseline for the whole pipeline. Considering the characters extracted from the text, a knowledge graph is created based on the sentiment between each pair of characters. Thus, if we have a poor character extractor, the following stages will suffer. Because of this, for our character extractor, we use a hybrid of two approaches:

- Model-based approach
- Rule-based approach

For the model-based approach, we tried two pre-trained NER models. The first one was spaCy [5], which uses a machine learning-based approach for NER, where it trains a statistical model on a large labeled dataset of text. The model is then able to recognize and classify named entities in new text based on patterns and features learned during training. The second one is Stanza [6], formerly known as Stanford CoreNLP. Stanza uses a rule-based approach for NER, where it first identifies potential named entities in the text using rules that are defined based on features such as capitalization and part-of-speech tags. Then, it applies additional rules to classify the identified entities into predefined categories such as person, organization, location, and others. Out of the two models, Stanza outperformed spaCy in every aspect: pre-trained, fine-tuned, trained from scratch.

For the rule-based approach, we use our own set of predefined rules for detecting common characters in folktale stories using regular expressions [7]:

- Animals, such as: wolf, lion, fox;
- Relatives, such as: mother, father, godmother;
- Characters, such as: prince, princess, king;

Furthermore, another pattern is searched using the regex form:

```
\b(?:the\s)?[A-Z][a-z]+\s[A-Z][a-z]+\b
```

The pattern searches for one or two words starting with capital letters, with a possible ’the’ before them. For example, the pattern detects ’the King’. This pattern is added for detecting unexpected character types that are not in the predefined lists. Finally a threshold is used, for which if an extracted possible character is mentioned more times, then it is considered a character in the story. The threshold is applied to remove possible mentions of characters that don’t really interact in the story or other outliers.

The model-based approach using a NER model detects names of people really well. However, sometimes it fails to detect characters like animals, relatives and other of the sort. For this reason, we use a combination of both model-based and rule-based approaches, returning a union of both methods.

```
{'predicted': ['cinderella', 'sister', 'charlotte', '
    fairy', 'godmother', 'prince', 'king'],

'ground truth': ['cinderella', 'stepmother', 'sisters',
    'godmother', 'prince', 'king', 'queen']}
```

Listing 2. Comparing our character extraction with the ground truth.

We can see that most of the characters were detected, especially those that the story is focused on. However, we can see that the characters ’stepmother’ and ’queen’ weren’t detected, and that is because they were mentioned only 1-2 times in the whole story thus failed the threshold filter. Furthermore, we can see an additional character that was detected ’charlotte’, which is one of the sisters in the story. However, when annotating the story, ’charlotte’ was omitted because she is considered to be ’sisters’. Also, ’fairy’ is detected, which again is the ’godmother’ in the story. Even though it failed on some small aspects, we can see that the majority of characters that are more important were detected, with additional repetition of characters under different names.

Coreference resolution (CR)

In order to improve the performance of the sentiment analysis and to reduce the ambiguity of our data, we performed coreference resolution. This NLP operation has for a task the detection and linking of all the expressions in a text that refer to the same character. In other words, it is the process of determining which antecedent ambiguous pronouns or noun phrases (such as "he", "she", "it", or "they") they relate to in the text e.g. "Cinderella" and "she" refers to the same person.

In order to find the coreference resolution model that suits our task the best, we implemented a couple of them.

To begin with, we used the coreference resolution model provided by Stanford CoreNLP [8].

However, this module is not particularly excellent at resolving coreferences, more specifically at cluster head selection - a mention that would replace every coreference in the text.

Due to this Stanford CoreNLP deficiency, we then implemented the AllenNlp [9] coreference resolution model. However, loading this model uses a lot of RAM and GPU memory so it crashed each time we ran it. We also tried using Google Colab because of its resources but nevertheless, it still crashed.

Finally, we found a coreference resolution model on Huggingface which we've successfully implemented and gives satisfactory results. Fastcoref [10] is a Python package for quick, precise, and easy-to-use English coreference resolution. The pip-installable package offers two operating modes: a precise mode based on the LINGMESS architecture that offers cutting-edge coreference accuracy and a significantly faster model, F-COREF. In addition to the models that are already provided, the package offers a unique SpaCy component that can be plugged into a SpaCy(V3) pipeline. This is what we've used for our implementation. Figure 1 shows an example of how the package works.

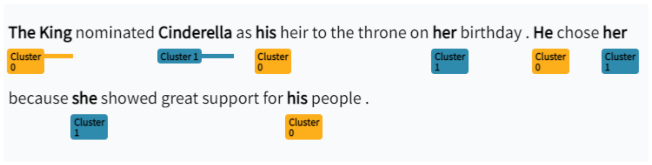
The King nominated Cinderella as his heir to the throne on her birthday . He chose her because she showed great support for his people .

Figure 1. Coreference resolution example of the Fastcoref model.

Sentiment analysis

Sentiment analysis is a method that consists of analysing a text in order to identify the emotional tone or attitude that the author is expressing. The objective of sentiment analysis is discovering the polarity of a sentiment, which may be negative, neutral or positive. Our main goal while using sentiment analysis was to identify the sentiment of the characters' relationships in a given story.
The first method we tried is Afinn [11]. Despite of its simplicity Afinn, which is a 3300+ words lexicon is an extremely popular method for sentiment analysis. The lexicon uses a pre-defined list of terms where each term has a corresponding sentiment score (integer) in the range between -5 (negative) and +5 (positive). One of Afinn's limitations is the fact that it is based on unigrams (single words), meaning that it only considers the sentiment scores of individual words and ignores the context or syntax of the text. In instances in which the sentiment of a sentence or document is affected by elements like sarcasm, irony, or negation, this may produce inaccurate results. One such example is the sentence:

"I am not having a good day."

Clearly, this sentence has a negative sentiment. However as seen on listing 3, because it solely relayed on the sentiment of the word "good", Afinn's verdict of this sentence is POSITIVE.

```
{
  "verdict": "POSITIVE",
  "score": 3,
  "comparative": 0.42857142857142855,
  "positive": [ "good" ],
  "negative": []
}
```

Listing 3. Afinn's verdict of the sentence "I am not having a good day".

Some sentiment analysis methods attempt to determine the sentiment of a sentence as a whole by looking beyond just unigrams. If we consider the previous sentence example, they try to understand that because of the negation, its sentiment is a negative one. One of those sentiment analysis methods is our second approach, Stanford's Stanza [6]. Stanza is an open-source Python library containing pre-trained models for numerous NLP tasks, one of which is sentiment analysis. The sentiment analysis model is based on a deep neural network architecture and is trained on a large corpus of annotated data. The model labels each sentence with a value equal to 0, 1 or 2, which represent a negative, neutral, or positive sentiment, respectively.
Our approach to determining the sentiment of a character's relationship is the following: first, we set an offset, which determines the range of sentences we are going to consider. For example, if the offset is three, that means that we are considering the three sentences prior as well as the three sentences after the current one. We call these sentences our search area. Then, for every sentence in the text, we determine the search area based on the given offset. If a pair of characters is mentioned in it, then we calculate the dominant sentiment and add it to a dictionary. The dominant sentiment is calculated by counting the number of appearances of all present sentiments and then multiplying them with pre-defined weights. In the end, we have a dictionary for every character with all its relationships, in the form shown on Listing 4.

```
{'charlotte': [['cinderella', -1]], 'cinderella': [['
    fairy', 0], ['godmother', 0], ['king', 0], ['prince
    ', 0], ['princess', 0]], 'fairy': [['godmother', 0]
    ]}
```

Listing 4. Example of a dictionary containing the final character relationships.

A. Determining the protagonist and antagonist

In addition to determining the character relationships, we used sentiment analysis to determine the character sentiment, or, in other words, to determine the protagonist and antagonist. Since this was not the primary focus of the report, we didn't use elaborate methods, but we utilized the fact that Afinn gives sentiments from -5 to +5. We used the same method for determining the search area as for the sentiment analysis of a character's relationship. As for determining the sentiment, we summed the sentiment scores of all appearances of the character and we multiplied it with the total number of appearances. This was done for every character. Finally, the protagonist was declared as the character with the highest sentiment, whereas the antagonist was the character with the lowest sentiment.

Results

A. Character extraction (NER) resutls

Fairytale	Correct characters	Found characters	Ground-truth characters
Cinderella	4	7	7
Hansel & Gretel	6	7	6
Little Red-cap	3	6	5
Sleeping Beauty	5	6	10
Rapunzel	2	2	5

Table 1. NER results.

From table 1, we can see that in some stories, our character extractor worked really well. There are also instances in which additional characters were returned, as we can see in the story "Hansel & Gratel". However, most of the time, they are characters under different nicknames. Also, there are some instances in which the extractor performed poorly, as with the stories "Sleeping Beauty" and "Rapunzel", detecting around half of the characters. From the examples, we see the robustness of the character extractor in more complex stories containing more complex characters, failing to recognize where "fairy" and "old fairy" are two different character instances.

B. Sentiment analysis

Fairytale	Offset	Correct rel.	Found rel.	Ground truth rel.
Cinderella	0	1	7	13
	1	1	7	13
	2	1	7	13
Hansel & Gretel	0	4	14	20
	1	6	14	20
	2	4	14	20

Table 2. Sentiment analysis results using affin (no coreference resolution).

Fairytale	Offset	Correct rel.	Found rel.	Ground truth rel.
Cinderella	0	1	13	13
	1	1	13	13
	2	1	13	13
Hansel & Gretel	0	3	10	20
	1	3	10	20
	2	4	10	20

Table 3. Sentiment analysis results using affin.

Fairytale	Offset	Correct rel.	Found rel.	Ground truth rel.
Cinderella	0	1	5	13
	1	1	13	13
	2	1	15	13
Hansel & Gretel	0	3	12	20
	1	2	15	20
	2	4	17	20

Table 4. Sentiment analysis results using stanza (no coreference resolution).

Fairytale	Offset	Correct rel.	Found rel.	Ground truth rel.
Cinderella	0	1	8	13
	1	1	12	13
	2	0	12	13
Hansel & Gretel	0	0	9	20
	1	1	14	20
	2	0	16	20

Table 5. Sentiment analysis results using stanza.

In tables 2, 3, 4, 5, we compare our sentiment analysis models Afinn and Stanza with and without using coreference resolution. We can see that for both models, using coreference resolution returned worse results. Also, using a higher offset in most cases gave yielded better results. It is worth mentioning that a lot of the sentiments that did not match the ground truths were mostly because of the subjective annotation from us. In most cases, the difference was from neutral to negative or positive and vise versa, where the sentiment between the characters is questionable.

Visualization

Putting it all together, we use the python library NetworkX [12], for visualizing the final results of our pipeline. We construct a graph where the nodes are the characters extracted from the story and the edges are the interaction and sentiment between them. The edges are colored based on the sentiment: red (negative), gray (neutral) and green (positive). Also, the size of the node is dependent of its degree. The more the character is connected to other characters, the bigger the node is. This way it's easier to see, which characters contribute more to the story, by interacting with many other different characters. In figure 2,

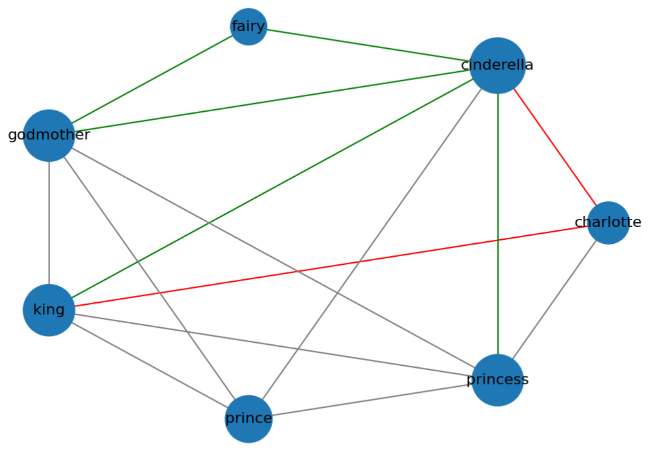

Figure 2. Predicted knowledge graph example1.

we can see the predicted knowledge graph for "Cinderella" short story, where Stanza is used throughout the pipeline for both character extraction (alongside rule-based approach) and sentiment analysis. We

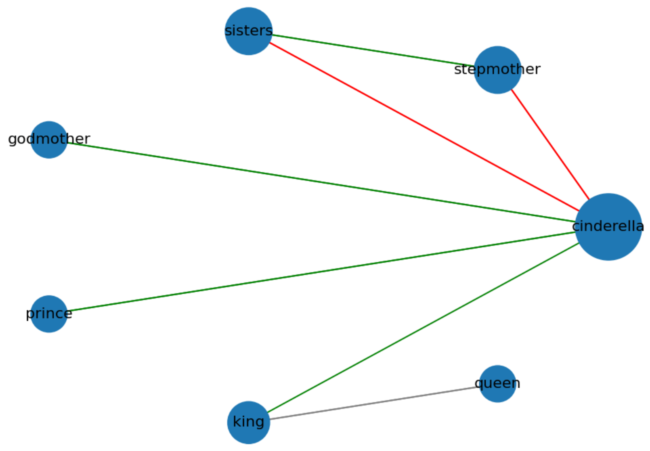

Figure 3. Ground truth knowledge graph example1.

can see that the predicted knowledge graph differs from the ground truth. By judging it strictly from the ground truth as we did in the tables above, our pipeline doesn't yield very good results. However, we can see that the majority of the characters were correctly detected, with a few missing and one additional. The character "charlotte" is indeed a character in the story. However, we omitted her from the annotation as she is one of the sisters. From this example, we can see the effects of the subjective annotation. Furthermore, we can see that the graphs also differ in respect with the sentiments. However, the sentiment in which they differ the most is neutral, thus it doesn't

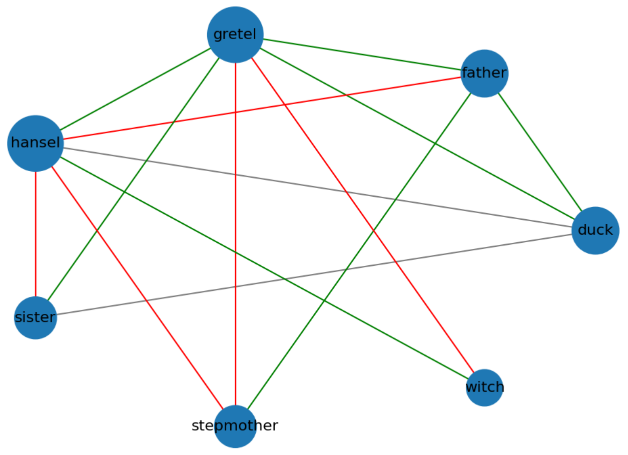

Figure 4. Predicted knowledge graph example2.

pose that much of a difference. In figure 4, we can see the predicted knowledge graph for "Hansel & Gretel" short story. In this example, for the sentiment analysis we used Afinn. In this example, we can see that the predicted knowledge graph resembles the ground truth much better. We can see that all characters were correctly detected with just one additional character "sister". Furthermore, we can see that the sentiments are way better here than in the previous example, where most of the important character relations are correctly detected.

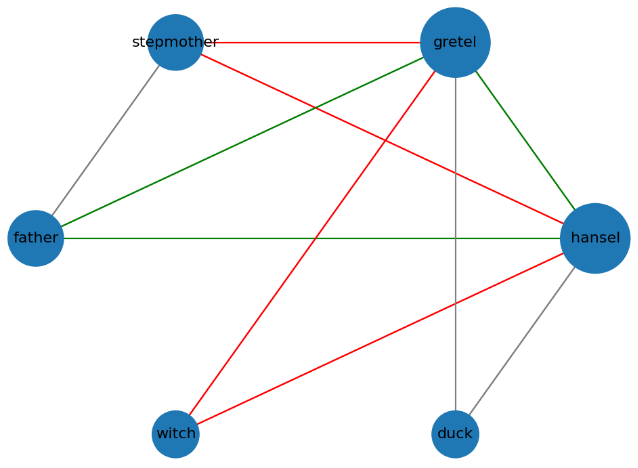

Figure 5. Ground truth knowledge graph example2.

Overall, the predicted knowledge graphs capture the relation between the majority of the characters and give a good insight to short stories. The results for sentiment analysis are weighted based on the type of sentiment, where for positive or negative sentiment the weights are higher (0.5 and 0.3) then for neutral (0.2). The weights were chosen empirically, from which using a higher weight for positive sentiments yielded better results in most cases. Also, the sentiments were calculated without coreference resolution, as we saw from the tables above, in most cases using coreference resolution yields worse results.

0.0.1 Determining protagonist and antagonist

To wrap things up, we briefly touched upon the topic of determining the protagonist and antagonist of the story. As it was explained earlier in the report the model we used was Afinn and the offset used was 0. Different combinations were tried but this combination gave the best results. Table 6 shows that the obtained results were sufficient, taken into consideration as determining the protagonist and antagonists wasn't our main goal for this seminar.

Fairytale	Cinderella	Hansel & Gretel
Protagonist	cinderella	gretel
Groundtruth protagonist	cinderella	hansel, gretel
Antagonist	charlotte	witch
Groundtruth antagonist	stepmother	witch

Table 6. Protagonist and antagonist results.

Conclusion

In this project, we tackled with various NLP tasks, implementing a knowledge base by using various methods and techniques for character extraction, coreference resolution and sentiment analysis. For the character extraction, we combined model-based and rule-based approaches. For the coreference resolution, we tried out several models, of which we landed on Fastcoref because of RAM shortcomings. For the sentiment analysis, we tried out two models and different sentiment scoring, such as adding weights to the sentiments of the models and also offset, analysing the sentiment from different search areas. Finally, using the results, we build a graph displaying the predicted knowledge graph with characters as nodes and their sentiments as the edges.

Overall, the results look quite promising and they can be further improved by also exploring other techniques and methods. Also, the results could be improved by annotating more data to which the models can be fine-tuned.

References

[1] 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).

[2] Alisha Bajracharya, Saurav Shrestha, Sharmila Upadhyaya, Bk Shrawan, and Subama Shakya. Automated characters recognition and family relationship extraction from stories. In *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 14–15, 2018.

[3] Project gutenberg. Accessed on April 27, 2023.

[4] Grimms' fairy tales by jacob grimm and wilhelm grimm. Accessed on April 27, 2023.

[5] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[6] 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.

[7] Alfred V Aho. Algorithms for finding patterns in strings, handbook of theoretical computer science (vol. a): algorithms and complexity, 1991.

[8] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. 01 2014.

[9] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

[10] Shon Otmazgin, Arie Cattan, and Yoav Goldberg. F-coref: Fast, accurate and easy to use coreference resolution. In *AACL*, 2022.

[11] Finn Årup Nielsen. A new ANEW: evaluation of a word list for sentiment analysis in microblogs. In Matthew Rowe, Milan Stankovic, Aba-Sah Dadzie, and Mariann Hardey, editors, *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, volume 718 of *CEUR Workshop Proceedings*, pages 93–98, May 2011.

[12] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.