University *of Ljubljana*
Faculty *of Computer and Information Science*

Fictional characters analysis

Anže Mur, Jaka Bernard, Edo Ljubijankić

Abstract

While analysis of literary works and their content is a commonly taught and often simple skill used by people, it is a challenge for machines. They lack human knowledge, common sense, and contextual awareness, which is very important when analyzing literary works. Many researchers have tackled these problems, some more successfully than others. In our work, we explore a subset of literary analysis, focusing on fictional character analysis. We approach the problems of character extraction, sentiment analysis of character relationships, and protagonist and antagonist detection. All of these tasks are performed on our newly created and annotated corpus of fables.

Keywords

fictional characters analysis, named entity recognition, sentiment analysis, network analysis, coreference resolution

Advisors: Slavko Žitnik

Introduction

Literature is almost infinitely diverse and as such contains many characters, events, and variations of those. During education, young people are often made to understand the relationships and interplay amongst various characters, factions, and events in numerous literary works. This is usually not very difficult for people but can be a hard nut to crack for computers, as natural language contains ambiguities and unclear references that are difficult to understand for machines, but possible to parse apart from context clues for human readers. Among such difficult tasks is the analysis of inter-personal relations between fictional characters due to the shortages of suitable corpora that would allow for efficient research. With that in mind, we present a newly created and annotated corpus of fables. Additionally, in our work, we present our implementations for the extraction of the characters appearing in the annotated fables, detect their sentiments towards each other and attempt to extract the protagonists and antagonists of the fables. Our implementation of the mentioned methods and our newly created corpus are publicly available on our GitHub [1].

Related work

With natural language processing (NLP) being a hot research topic in recent years, there are many corpora dedicated to NLP tasks. As a result, there are many articles delving into various issues including literary analysis.

The authors of [2] present a new corpus of 50k crowd-funded five-sentence commonsense stories called ROCStories. This corpus is used in their evaluation framework Story Cloze Test which requires a system to select an appropriate right or wrong ending for the four-sentence tested story. The presented framework can serve as a generic story understanding and story generation evaluation framework. Another evaluation benchmark corpus is shown in the article [3] with the focus on the evaluation of reconstruction of event-driven plot structures.

The authors of [4] present an ACE entity annotated dataset, collected from 100 different English literary texts obtained from Project Gutenberg. Corpus contains 210,532 tokens that are categorized into six different categories - person, organization, location, geopolitical entity, facility, organization, and vehicle. Additional work on the corpus was done in [5] which authors collected the annotations for the events. They also presented and evaluated the current SOTA models on the collected corpus.

Knowing relations between characters is an important part of story understanding and summarization. The authors of the article [6] present a method that identifies the main characters in a story and extracts relations between them. The proposed method is a hybrid approach that combines the features of unsupervised and supervised learning methods. For testing purposes, 100 short stories for kids were used and 300 char-

acter pair relations were analyzed. The system identified if the relation between pair of characters is 'Parent-Child' or 'Friendship' or if there is no relation found. The proposed method proved to be good compared to other existing methods.

In article [7] authors present linguistically informed deep neural network architecture for automatic extraction of cause-effect relations from text documents. The proposed architecture uses word-level embeddings and other linguistic features for detecting such events. Bi-directional long short-term memory (LSTM) and other models were used and compared in experiments. For testing, multiple datasets were used. Most tests were run on SemEval 2010 annotated dataset, ADE or adverse drug effect dataset, BBC News Article dataset and Recall dataset.

Authors of [8] focused on the automatic creation of summaries from short stories with the objective to give users relevant information about the setting of the story without revealing the plot. That would help the user decide whether he should read a full story. They produced summaries by extracting information about important entities and events from the stories. While summaries are not of high quality they do achieve the original objective.

Dataset

We were not able to find any sufficient short story dataset that would have annotated characters and the relationships between them. This led us to a decision that we would create our own dataset and provide our own annotations for it. With the goal of creating a sufficiently large dataset on which the evaluation of our methods would be represented in a statistically correct way, we decided that the optimal genre of a short story would be a fable since its annotation would not take as much time.

Fables are short stories that feature animals, forces of nature, and inanimate objects that usually possess some human-like features. The interactions between the main characters usually result in some moral lesson. Usually, this moral lesson is repeated explicitly at the end of the story as the one-sentence summation of it. Most of the well-known western fables are attributed to the Greek author Aesop, who was supposed to be a slave in ancient Greece. Because we were familiar with a lot of his fables we decided to use a translated version of them as our dataset. We found a translated collection on the Project Gutenberg [9] website, which provides over 60 thousand free eBooks digitized by volunteers. The project is focused on older work, for which the U.S. copyright has expired so the texts are in the public domain and we are allowed to use them. The fables were translated by Joseph Jacobs and collected in a book called The Fables of Aesop [10].

To create an annotated corpus, we created two new files for each of the selected fables. In the first file, we saved the raw text of the fable and in the second one, we saved its corresponding annotations. For each of the fables we annotated:

- its characters,
- the sentiment relationships between the characters,
- the protagonist and antagonist of the fable.

To obtain this data we read and evaluated every selected fable. We saved the annotations in the JSON format and we can see an example of our annotation for the fable *The Sick Lion* in the Listing1.

Listing 1. Annotation for *The Sick Lion* fable in JSON format. In the "characters" field we can see the characters that appear in the fable. The "protagonist" and "antagonist" fields denote the protagonist and antagonist of the fable by their names. The "sentiments" field includes all of the characters by their names and specified sentiment relationships between them. We divided the sentiment relationships into three different classes: -1 as negative sentiment, 0 as neutral sentiment, and 1 as positive sentiment.

```
{
  "characters": ["lion","boar","bull","ass"],
  "protagonist": "lion",
  "antagonist": "ass",
  "sentiments": {
    "lion": {
      "lion": 0,
      "boar": -1,
      "bull": -1,
      "ass": -1
    },
    "boar": {
      "lion": -1,
      "boar": 0,
      "bull": 0,
      "ass": 0
    },
    "bull": {
      "lion": -1,
      "boar": 0,
      "bull": 0,
      "ass": 0
    },
    "ass": {
      "lion": -1,
      "boar": 0,
      "bull": 0,
      "ass": 0
    }
  }
}
```

As we can see from the listing 1, for each of the characters we evaluated the sentiment relationship between him and all of the other characters in the story. Sentiments relationships are described in three different classes:

- **-1**: negative sentiment,
- **0**: neutral sentiment,
- **1**: positive sentiment.

We set the sentiments of the characters with themselves to neutral. Overall we collected and annotated 55 different fables which are all publicly available in our GitHub repository [1].

Methods

In our work, we concentrated on three different tasks of fictional character analysis: character extraction, sentiment analysis of character relationships, and protagonist/antagonist detection. The methods used for each of the listed tasks are more thoroughly explained in the following sub-sections.

Character extraction

Character extraction is an integral part of our fictional character analysis since an accurate character extraction is a basis for future character analysis. Our character extraction pipeline is composed of two different stages: named-entity recognition and coreference resolution.

Named-entity recognition (NER)

We decided to use named-entity recognition which is used as an information extraction method in many language processing tasks. It classifies different detected entities in text into predefined categories.

For the actual task of NER, we decided to use two different pre-trained models. The first one is part of the Stanza open-source NLP tool [11] created by the Standford NLP Group. The model uses contextualized string representation sequence tagger with the LSTM language model. The model is trained two times separately using the forward and backward characters. At the tagging of the input text, the representations from each of the trained models are concatenated and the result is fed into a one-layer Bi-LSTM sequence tagger.

The second model that we used is part of the open-source NLP software called spaCy [12]. It uses a pre-trained RoBERTa [13] transformers-based model trained on a large corpus of English language data. The model uses self-supervised learning so it can automatically generate inputs and labels for raw text data. The model learns the inner representation of the English language so it can be used to efficiently extract features.

Both of the used models first tokenize the input text and then outputs the predictions for tokens that were detected as some kind of possible entity. Models can predict various different kinds of entities for example organizations, locations, events, products, people, etc. But since we want to extract only the characters from our stories we filtered the obtained entities to include only the entities with the person tag. After, we removed any possessive signs (ex. John's is transformed to John) and possible article words (the, an, a, and) from the obtained entities.

Coreference resolution (CR)

To further improve the performance we decided to perform the task of the coreference resolution over our input text before performing the named-entity recognition task. Coreference resolution is the task of mention (or linguistic expressions) clustering which is composed of two main stages: mention identification and mention clustering. Mentions are referring expressions that can be pronominal (ex. I, he, this, its, etc.) or nominal and can represent any subject. Identified mentions can be replaced with the selected mention that is the best representation of our entity. With the usage of coreference resolution, we can reduce the ambiguity of our text which serves as a great prepossessing pipeline for our NER task.

For the actual task of coreference resolution, we used a pre-trained model based on the article [14] which is part of the AllenNLP [15] open-source research library. The models tries to get the embedded representations of each span in the input text. The embeddings are created with SpanBERT and are scored with the goal of removing the spans that would unlikely occur in coreference clusters. The model then decides which of the remaining spans are coreferent with each other. For the tokenization of the input text, the model uses spaCy's tokenizer.

The used library is not very good at resolving the coreferences, more specific it is not very good at the cluster head selection. A cluster head is a mention with which every coreference in the text would be replaced and the used library just selects the first detected one. We changed that by ignoring the clusters that don't include any noun phrases and selected the head in a more profound way. For the cluster head selection, we tried two different methods: selecting the first noun phrase and selecting the most occurred noun phrase. The first one yielded much better results so we decided to use it in further experiments. After the head is selected each of its coreferences is replaced with it in the original text.

Our final character extraction pipeline now consists of the coreference resolution followed by the named-entity recognition. After the characters are extracted they are saved into a file for evaluation.

Sentiment analysis of character relationships

To obtain sentiment for character pairs, we first used our implementation of character extraction to obtain their names. We computed a sentence sentiment vector on the unedited text using two methods. The first one is Afinn [16], a wordlist-based sentiment analyzer that contains more than 3300 words with their corresponding polarity scores. For the second model, we again used Stanza's sentiment analysis pipeline, which is based on convolutional neural networks [17]. The created vector contains the sentiment of each sentence in the text. We then replaced the coreferences in the text with the relevant character names, after which we obtained a character occurrence matrix that contains the occurrences of each character in each sentence of the provided text.

After obtaining the occurrence matrix we computed the character co-occurrence matrix by multiplying the occurrence matrix with its own transposition, obtaining a $n \times n$ matrix, where n equals the number of detected characters. Each entry in the co-occurrence matrix represents the number of co-occurrences in the text of the characters connected to the relevant row and column of the matrix.

We also compute the sentiment matrix, which is done by multiplying the occurrence matrix with the product of its own transposition and the sentence sentiment vector. This results in a $n \times n$ matrix, where n equals the number of detected characters. Each entry in the sentiment matrix represents the sentiment of the pair of characters connected to the relevant row and column of the matrix. The matrix is then aligned by using the co-occurrence matrix and alignment rate variable, to account for the author's writing style.

Both the co-occurrence and the sentiment matrix have their lower triangle computed, after which the diagonals are set to 0, as characters do not have co-occurrences with themselves, nor do they have sentiments with themselves. After this the sentiment matrix is also normalized to an interval between -1 and 1 and saved to a file, to await evaluation.

Protagonist and antagonist detection

Among our analysis endeavors, we decided to attempt protagonist and antagonist detection. We took several different approaches with varying degrees of success.

For the first technique, we used a network analysis algorithm called PageRank [18]. The algorithm was designed by Google for use in the ranking of web pages in their search engine. It outputs a probability distribution representing the likelihood that a person will arrive at a particular node in a graph by randomly following edges between the nodes. It initializes the same probability for all nodes in the graph, then iteratively adjusts them towards the true value by transferring the PageRank of a given node to the targets of its outbound edges with a uniform distribution upon the next iteration. If the algorithm runs into a dead-end in its walk through the graph nodes, it jumps to a random node in the said graph and continues from there. For the algorithm to work we first had to create a network from our character sentiment matrices. Each node in the network is represented by a character and the edges between them are the sentiments of the characters. PageRank is then used to rank the nodes in our character network. In the next step, we extract the top character with a positive sentiment and the top character with negative sentiment and assigned them as the protagonist and antagonist, respectively.

The second technique we used was taking the co-occurrence matrix and sentence sentiment vector and multiplying them, then summing the matrix up by characters, thereby obtaining character sentiments. The co-occurrence matrix is of shape $m \times n$ where m is the number of sentences in a text and n is the number of characters detected in the text. For each sentence, the matrix has an entry of zeros and ones depending on whether a specific character is present in that sentence (1) or not (0). The sentiment vector is obtained by computing the sentiment of each sentence in the text. After multiplying the matrix and the vector we then took the entries with the best and worst sentiments and assigned them as the protagonist and antagonist.

The third technique we used was taking the co-occurrence matrix and counting the number of occurrences for each character, as we believed that a protagonist should appear the most often in a given text, while an antagonist might be in the second place due to the importance of their role to the story. We then traversed the resulting array by descending the number of occurrences and assigned the first character as the protagonist and the second as the antagonist.

For the fourth technique, we combined character sentiments and character occurrences, once again traversing the occurrence counts in descending order, but only assigning a character to a role if their overall sentiment matched that of the assigned role, so positive sentiment for the protagonist and negative sentiment for the antagonist. All protagonist-antagonist pairs found with the above techniques were saved in a separate file for each text, to await evaluation.

Visualization

From our sentiment and co-occurrence matrices, we generated character relation graphs. In the sentiment graph, positive sentiments are represented with lighter edge colors while negative sentiments are represented with darker edge colors. Node sizes are based on the number of character occurrences within the relevant fable. In the co-occurrence graph, lighter edge colors represent fewer character co-occurrences, while darker colors represent more character co-occurrences. We generated visualizations of these graphs for each fable.

Results

Character extraction

We performed character extraction with both of the previously described models with or without the usage of coreference resolution (CR). For the evaluation, we calculated the precision, recall, and F1 scores for each of the used models, compared to our ground truth annotations. We can see the performance results in Table 1.

Table 1. Character extraction results.

Used model	Precision	Recall	F1 score
spaCy	0.96	0.36	0.41
spaCy + CR	0.96	0.39	0.45
Stanza	0.87	0.64	0.67
Stanza + CR	**0.88**	**0.70**	**0.73**

As we can see from the obtained results Stanza's model outperforms spaCy's model in terms of F1 scores and recall by quite a lot. By further inspecting the results spaCy's model outputs much fewer entities in general compared to Stanza's model. We can also see that the usage of coreference resolution greatly impacts the performance of both of the used models. The reason behind this is that the character names get repeated multiple times in our preprocessed text and are more likely to be detected as named entities by NER.

Sentiment analysis of character relationships

Once we computed and saved our predicted character sentiment, we filtered the fables where our annotations contained at least every character detected by our named-entity extractor. We then computed the precision, recall, and F1 score metrics between our ground truth annotations and extracted sentiments for all overlapping characters between the two files. We then averaged the obtained precision, recall, and F1 scores over all eligible files. Of the 55 annotated fables, 40 fit our evaluation criteria. The results can be seen in Table 2, where we can see that the Afinn analyzer performs better than the Stanza pipeline.

Table 2. Sentiment analysis results.

Sentiment model	Precision	Recall	F1 score
Afinn analyzer	**0.83**	**0.83**	**0.83**
Stanza's pipeline	0.79	0.79	0.79

Our visualizations are usually not very informative with so few characters in each fable, but they are quiite interesting in a few cases. As we can see in the sentiment graph in Figure 1, all of the sentiments between the characters are very positive, but the cat-venus and cat-jupiter edges are not as yellow as the others. Their green color indicates that these sentiments are not as positive as the rest, even though they are still positive overall. When looking at the co-occurrence graph in Figure 2, one might think that there is no connection between the cat and the two gods, but looking closely we can see, that the connections are very light, meaning there are very few co-occurrences between these characters in the fable.

Protagonist and antagonist detection

Once we performed the sentiment analysis of character relationships using our two selected models, we observed that Stanza's model performed significantly worse than the Afinn analyzer, which is why we decided to use Afinn for our protagonist and antagonist detection. For this evaluation, we compared our model's detected protagonists and antagonists against their annotated counterparts by computing the detection accuracy. We did so separately for protagonists and antagonists, respectively. The results of our detection can be seen in Table 3.

Table 3. Protagonist and antagonist detection results.

Detection method	Protagonist acc.	Antagonist acc.
PageRank	0.35	0.20
Char. sentiment	0.27	**0.27**
Char. occurrence	**0.70**	0.24
Char. occur. + sent.	0.25	0.24

As we can see, character sentiment is the best in antagonist detection by a small margin, while the character occurrence detection method is by far the best for protagonist detection among our chosen methods. The worst for protagonist detection is the character occurrence and sentiment method, while PageRank is the worst for antagonist detection.

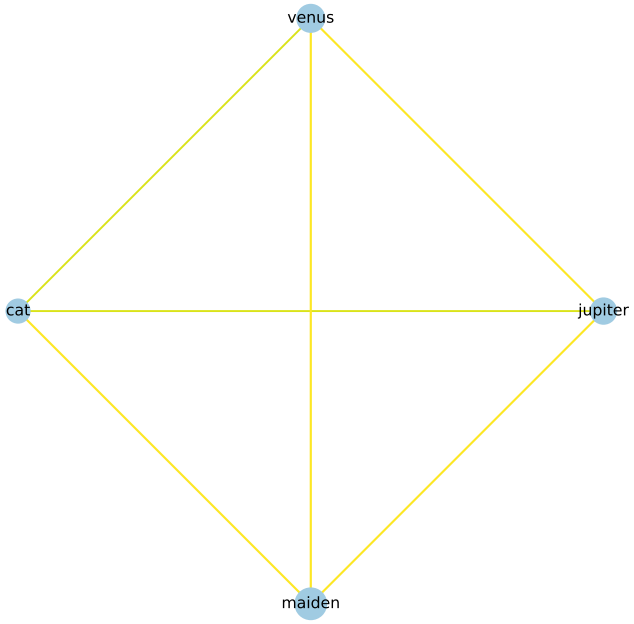

Figure 1. A character sentiment graph visualization. The sentiment visualization of *The Cat Maiden*. Lighter shades mean better sentiment, so the sentiments of cat-venus and cat-jupiter are worse than the rest but still positive.

Discussion

We are quite satisfied with the results of our character extraction pipeline, especially after the addition of the coreference resolution. During our work we have encountered a problem that could be addressed in the future - the performance of various NER solutions, as no solution perfectly detects all entities participating in a certain text. This results in missing or additional characters, which also impacts our sentiment detection results as good character detection is needed for character occurrence computation. Both of the used NER models would often recognize capitalized words as person-type entities. For example, the phrase "Good-bye" is recognized as a person in the fable *The Town Mouse and the Country Mouse* because it is capitalized. But the processing of all lower case text yields worse results since it looks like capitalization is an important part of the entity recognition.

While our sentiment analysis performed admirably, with the Afinn analyzer achieving 0.83 in all three scores, it is lacking in several aspects. Using our method of computation assumes that the sentiment between two characters is equal in both directions, and does not take into account the possibility that characters might have different sentiments towards each other. A possible solution for this could be including the subject and object of a sentence in the inter-character sentiment computation, so the sentiments would be computed one-way

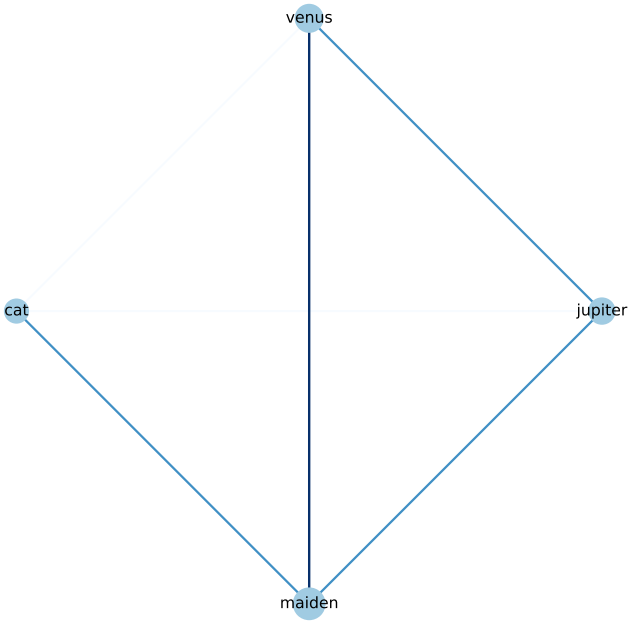

Figure 2. A character co-occurrence graph visualization. The co-occurrence visualization of *The Cat Maiden*. Darker edge colors mean more co-occurrences between two characters. The pairs cat-venus and cat-jupiter have very few co-occurrences, so the two edges are very light whereas the venus-maiden pair have many co-occurrences so the edge is of darker color.

only if the two characters occupy the subject and object roles in a sentence. In shorter texts such as fables, this might introduce the problem of lacking sentiments, as there are not many sentences and therefore not many chances for character pairs to appear in the text.

Our protagonist and antagonist detection did not perform very well. While we managed to reach a 0.47 accuracy in protagonist detection with the character occurrence method, the rest of the results are rather poor. To improve this, we would need to determine better scoring criteria for the characters, to help us determine their roles in their story.

There are also differences between what a reader or annotator perceives including the contextual information available to them and what the algorithm is capable of perceiving. A reader usually knows that a fox is stereotypically sly and cunning, while a lion is usually prideful, domineering, brave, and noble, and grandparents are usually old and wise. For example, in the fable *The Fox and the Crow*, the fox compliments the crow, but also steals from it, which may be perceived as a positive sentiment by the fox due to the compliment, but a reader knows is negative due to the theft. An algorithm usually has no prior knowledge and therefore lacks the context the annotators use to decide sentiments between characters. There is also non-mentioned information, which can be inferred by the reader, like gathering that a hunter was able to catch and kill their prey from a sentence such as ”The hunter collects his trophy”. The reader may also perceive changes in sentiments through time in a text, which is reflected in neither the annotations nor the results of our sentiment estimation. We also encountered issues with character extraction. The first was when certain characters may be referred to by obscure words, contextual information, or common sense knowledge that is usually not available to an algorithm. An example of such knowledge can be seen in the fable *The Fox and the Cat*, where a cat is referred to as ”Miss Puss” at some point, which a human reader is likely to associate with the word ”pussycat” which is another word for cat. While a reader would know this, our algorithm misses this and identifies ”Miss Puss” as another, new character in the fable. Another example is the issue of parsing important contextual information, as seen in the fable *The Ass in Lion's Skin*. An ass disguises himself using a lion's skin, an object, to appear like a lion. No lion appears in the fable, yet a lion is found as a character due to the appearance of its skin.

Conclusion

We tried various methods for character detection, sentiment estimation, and protagonist and antagonist detection, with varying degrees of success. We encountered some common hurdles in natural language processing, like co-reference resolution and finding the correct named entities. Various aspects of our implementation could be improved by using better individual components like the model responsible for named-entity recognition or the scoring function for protagonist and antagonist detection. Having used fables, we do not know how our solution would perform on longer texts, but it would likely present challenges associated with longer literary works.

References

[1] Fictional characters analysis. https://github.com/anzemur/literacy-knowledge-base. Accessed: 18-03-2022.

[2] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet 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, 2016.

[3] Tommaso Caselli and Piek Vossen. The event storyline corpus: A new benchmark for causal and temporal relation extraction. In *Proceedings of the Events and Stories in the News Workshop*, pages 77–86, 2017.

[4] David Bamman, Sejal Popat, and Sheng Shen. An annotated dataset of literary entities. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2138–2144, 2019.

[5] Matthew Sims, Jong Ho Park, and David Bamman. Literary event detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3623–3634, 2019.

[6] 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).

[7] Tirthankar Dasgupta, Rupsa Saha, Lipika Dey, and Abir Naskar. Automatic extraction of causal relations from text using linguistically informed deep neural networks. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 306–316, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[8] Anna Kazantseva and Stan Szpakowicz. Summarizing short stories. *Computational Linguistics*, 36:71–109, 03 2010.

[9] Project gutenberg. https://www.gutenberg.org/. Accessed: 18-03-2022.

[10] The fables of aesop by aesop. https://www.gutenberg.org/ebooks/28. Accessed: 18-05-2022.

[11] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*, 2020.

[12] Industrial-strength natural language processing. https://spacy.io/. Accessed: 20-05-2022.

[13] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[14] Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*, 2018.

[15] Allennlp. https://allenai.org/allennlp. Accessed: 29-04-2022.

[16] 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.

[17] Yahui Chen. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo, 2015.

[18] Facts about google and competition. https://web.archive.org/web/20111104131332/https://www.google.com/competition/howgooglesearchworks.html. Accessed: 03-04-2022.