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

Literacy situation models knowledge base creation

Mila Marinković, Ilina Kirovska and Radoslav Atanasoski

Abstract

Introduction

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 our 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 identifying the protagonists and antagonists.

Related work

In 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 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 the some relationships, new relationships are resolved using the propagation rules for relationship. Considering that character extraction accuracy equaled to 95.33% and relationship extraction accuracy to 80.5% it is evident that this approach was effective.

Dataset

Because we weren't 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 Grimm 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. Beside 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 txt file containing the raw text of the story and a json file

which 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

An example of an annotation file can be seen 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": ""
}
```

Methodology

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

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

Coreference resolution (CR)

In order to improve the performance of character extraction 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.

For the purpose of this project, we used the coreference resolution model provided by Stanford CoreNLP [5]. This module detects both explicit ("Cinderella" and "she") and implicit ("The Little Ash Girl" and "she") coreferences. Furthermore, it produces a set of clusters, with each cluster representing all expressions in the text that refer to the same entity. For example, in the Cinderella story clusters would be:

- Cinderella: she, Cinderella, the poor girl, she, beautiful lady
- Prince: he, him, the Prince's, his
- Stepsisters: Cinderella's stepsisters, her stepsisters, the stepsisters

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 deficiency, coreferences resolution represents one of the improvements for the final part of this project.

Fugure 1 shows how this module works in the background of our project.

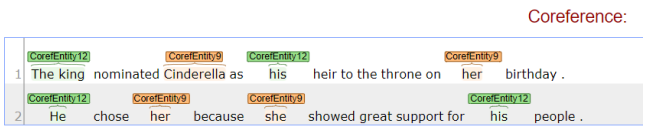

Figure 1. CoreNlp coreference resolution example.

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 [6], 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 [7], 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 [8]:

- 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 list. 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.

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 [9]. 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 [7]. 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. 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.

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/or antagonist if there is one. Our approach to detecting character sentiment was similar to the one used for character relationships, with the key difference being that we focused exclusively on mentions of a single character within the text.

Visualization and results

Putting it all together, we use the python library NetworkX [10], 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 sentiment
- Gray - neutral sentiment
- Green - positive sentiment

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

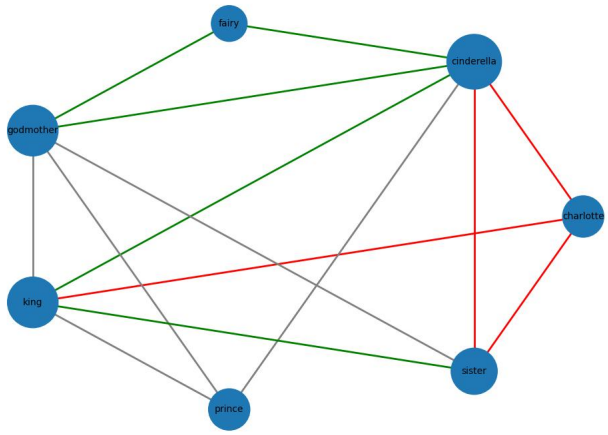

Figure 2. Predicted knowledge graph example.

figure 2, 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. The final results look very good in compari-

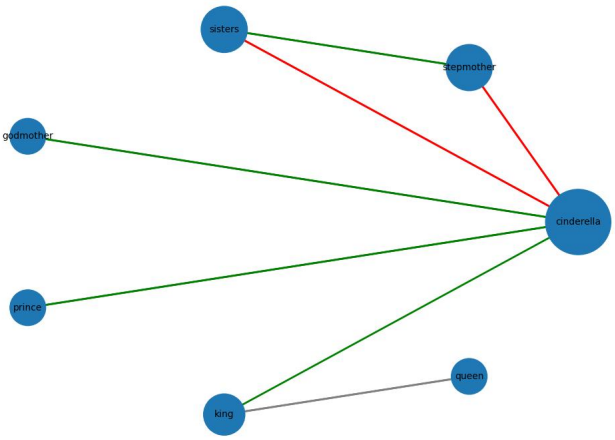

Figure 3. Ground truth knowledge graph example.

son to the ground truth from figure 3, correctly predicting the relation between 'cinderella', her 'sisters'

and her 'godmother'. However, there are still some mistakes coming from having additional characters and misinterpreting the relation between the 'prince' and 'cinderella'. Overall, the predicted knowledge graph captures the relation between the majority of the characters. The results for sentiment analysis are weighted based on the type of sentiment, where for positive or negative sentiment the weights are higher then for neutral. The results could be further improved by improving the weights, and also considering the global sentiment of the story.

Furthermore, these results are with the exclusion of coreference resolution from the pipeline. Thus, sentences containing 'her' instead of 'cinderella', and other similar ones were not taken into account, disregarding a big part of the text for evaluating the characters relationships.

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

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

[7] 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.

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

[9] 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.

[10] 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.