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

Character analysis in Slovene short stories

Gal Petkovšek, Marija Marolt, Jana Štremfelj

Abstract

One of the subfields of natural language processing is knowledge extraction from text and building knowledge bases from this data. In this work we focus on character analyses, more specifically character character detection, sentiment analysis and relationship extraction. We focused only on Slovene stories as there are not many similar tools for the Slovene language. We develop a workflow that extracts characters from the story using the classla NER model and some rule based approaches, then detects the links between characters and classifies them using the Slovene sentiment lexicon. We also create a dataset and trained a model for character sentiment classification.

Keywords

Natural Language Processing, character analysis, knowledge base, character relations

Advisors: Slavko Žitnik

Introduction

Natural Language Processing (NLP) is an evolving field trying to automatically extract the knowledge from the abundant volume of natural language text. Machines can process significantly larger amounts of text, but there are some challenges such as ambiguity of the text, the sentiment analysis, commonsense reasoning, information extraction etc.

The main task of our project is the knowledge base creation, which is a sub-field of Natural Language Processing. This (amongst other things) includes extraction of relations between the characters are created, classification of each character as positive or negative and the protagonist detection.

The pipeline and models are adjusted to Slovene language. We get a graph from the information extraction pipeline in which the nodes represent the characters and links the relations between them. First step in the pipeline is name entity recognition. In more general case this is refereed to recognition of all entities is made (e.g. person, location, organization...) however we only focus on characters. In the next step coreference resolution is applied where all expressions that refer to the same entity are found. Next, additional information is obtained from the text about the characters or the relations between them. In the last step the knowledge graph is constructed.

The visualization represents the relations of the characters and their profiles, which we obtain from the knowledge graph. It provides a visual depiction of the story structure and character interactions, which helps the reader quickly establish the general relations between characters.

Related work

Character analysis and relationship extraction is already a well researched topic that falls under the category of knowledge base extraction from text. Different authors report different approaches to tackling the problem of knowledge base creation, while most of them still roughly separate the pipeline steps to the ones meant to extract nodes ans the ones to extract edges. In an article by Kertkeidkachorn *et. al.* [1] they describe a pipeline for knowledge graph construction using with five steps: entity recognition (and mapping), coreference resolution, triple extraction, triple integration and predicate mapping. In another article [2] the authors describe the an algorithm called Grapher that uses an encoder-decoder language model (sequence to sequence algorithm) for node extraction and for edge detection they tested two different approaches (LSTM/GRU and classification).

When we consider the character analysis problem (which is a sub-problem of the above described knowledge base creation problem). Different research is also being done in that area. The article [3] by Nalisnick *et. al.* for example predicts the relationships from characters on Shakespeare's plays, however their problem is different from ours in the way that since we have plays the problem is mainly focused on the dialog between characters. Another example of character analysis is described in the article [4] where the author's main focus is to classify the character as positive or negative which is done in

3 different ways: character's speech, character's actions and predicatives.

In Slovene however, the topic is not as well covered. Markovič [5] *et. al.* in his article for example uses network analysis for analysing Slovene fables, however it does not strictly cover sentiment analysis of characters. In another article [6] the authors describe how they prepared a Slovene database for the problem of classifying texts as positive, negative or neutral, however they do not focus on the characters in those texts. Therefore to the best of our knowledge not many authors have attempted to solve the character analysis problem purely in Slovene language.

Methods

Data preparation

The dataset that we have chosen, is a collection of Slovene short stories. Our training corpus contains 74 fairy tales. They are obtained from dLib.si [7], which is a digital library and document server for Slovenian language resources. Our objective was to identify stories with similar themes and characteristics that would be interesting for analysis. These stories were specifically chosen due to their length, which allows for efficient processing and label with the available resources.

The stories were scraped from the website using a Python web scraper, which utilized the BeautifulSoup and requests libraries. The obtained text was then processed by replacing multiple spaces with single spaces and removing new line breaks.

To train the model, the data was labeled with the assistance of ChatGPT, which aided in identifying the characters from the fairy tales and classifying their sentiment. These results were then reviewed and manually corrected, as it typically only provided a list of the main characters. Characters who appeared only a few times needed to be added manually.

Name entity recognition

Name entity recognition is in general the task of extracting different entities from the text (locations, people, organization...) which is usually done with specialized NER language models. Our goal was a bit more domain specific as we wanted to extract the names of characters of fairy tales. The problem that arose here is that NER models are usually trained to recognize names of people as person entity, which in fairy tales is rarely the case as most characters are not introduced by names (eg. king, queen, fox *etc.*).

We first tackle the problem by capitalizing the first letter of each character name which made the model perform very well, but since this usually meant that a human needs to manuly perform this task we abandon this idea as it defeats the purpose of automatised entity extraction.

In our second approach we tackle the problem in two ways and then combine the results. First we focus on the characters that are referred to with names. For recognizing those we use classla NER model. For recognizing all the other characters (which are the majority) we use chatGPT to generate a list of most common character names in Slovene fairy tales. This list enables us to recognize characters such as zmaj, oče, čarovnica in addition named characters such as Janko, Matka. We assumed that there is a somewhat small set of characters that appear in the stories so it is possible to capture and list them all.The list was then also manually corrected (some characters were removed and some were added). We then use this list to identify characters. However because of different shapes of words we do not just search over the text but rather over the lemmatized words of the text, which is obtained from the classla model. After some initial testing we also apply Levenshtine distance to compare the words as the lemma model sometimes fails (however in the large majority of cases it works very well). We also use the Part-of-speech (POS) model from classla to confirm that the found word is indeed a noun.

Additionally filtering is added as the component predicted some characters incorrectly (false positives). We therefore also filter out the characters that do not have many occurrences as the stories usually do not have characters that would just appear once.

The result of this module is a dictionary of all occurrences of all characters (their positions in the text) in the text, an example of which can be seen below

```
{
    "zmaj": [[54, 59],
             [147, 151],
             [188, 193],
             [558, 562]],
    "pastir": [[339, 345], [107, 116]],
    "ovca": [[474, 478]],
    "zver": [[259, 263],
             [121, 125],
             [314, 318]]
}
```

Coreference resolution

For the coreference model we chose the SloCOREF model (Coreference Resolution for Slovene language). It returns pairs, with the trustworthy value of the prediction of the model. Each element of the pair includes the starting and ending position in the text. From the Clarin site [**?**] we downloaded the model and we used some of the code from the GitHub site [**?**].

We then use the entity dictionary from the name entity recognition component and the coreferenced pairs (nouns and pronouns) from the coreference model. If one element from the coreferenced pair is recognized as the person entity, the other start and end position is added to the list of the entity occurrences. If none of them is a person entity, but one is recognized as entity, we add the other occurrences to the list. If both are entities we add the second start/end position to the first list of occurrences.

Because the model cannot take the whole story at once, we spilt the story to parts and search for the correference pair

on these parts with certain offset. For example the length of subtexts is 1000 characters and the offset is 500, that means for the first part we take first 100 characters and for the second part from 500 to 1500 etc.

The output of the function is the modified dictionary of the entities and its starting/ending positions, just as in the previous step.

Relationship extraction and sentiment classification

The relations are searched in using the output of the entity recognition component. We predict that the two entities are connected if they appear in the same sentence. Link classification is made with the help of Slovene sentiment lexicon[8]. The verbs from the sentences (detected with classla's POS model), where two entities occur together, are classified with the lexicon. For each verb, the weight between -5 and +5 is assigned. The lower the weight, the negative the relation. The weights of verbs from the lexicon are summed together and averaged over all occurrences of a pair.

Character properties extraction

From entity occurrences, we acquire the characteristics of entities by extracting adjectives (detected with the classla's POS model) around the first few occurrences of an entity (as in fairy tales the character is very vividly described when it is first introduced). For each entity a list of adjectives is returned.

Character importance

The importance of a character is based on the number of times it shows up in the story and on the weight from the pairs. The weights from pairs is defined by the importance of a link between two entities (the number of times they appear together in a sentence). A dictionary of all characters and its normalized value (importance of a character scales form 0 to 1) is returned.

Character sentiment analysis

Furthermore we also classify the sentiment of each character in the stories. We tackle this problem by taking the occurrences of the detected entities in the text, extracting contexts and then classifying them. Since we were doing the learning on the Slovene stories we did not have any labeled data for this problem. For or the labeling of the character sentiment we prompted ChatGPT (prompt can be found in the code on github). We tried to predict three classes (positive negative and neutral), however ChatGPT did not produce very consistent results for such classification. Therefore we decided to simplify the task and only have positive and negative labels (1 and 0). After the sentiment labels the character locations also needed to be labeled. ChatGPT did not produce good labels for the positions of entities as they were incomplete and in most cases even incorrect. Therefore for that we used the output of the entity recognition component, which was then checked and expanded by hand.

After the labeling we created different datasets for training the models. When constructing the samples we use a similar method as is described in the article by Žitnik *et. al.*. For every occurrence of the character in a story we extract also a certain number of words before and after the character. We then concatenate all those contexts into a single string separating them with a $ character and adding the sample the sentiment label of the character.

For the modeling we use the destilbert_base_cased model for classification as it is smaller than regular bert and a custom model that used an embedding layer to embed the string and multiple convolutional layers to extract features. The models output a float value which is for the evaluation thresholded at 0.5 (anything above is good and below is bad).

Visualization

Purpose of visualization is to enables a comprehensive understanding of character relationships, sentiment, importance, and descriptive attributes. It was created using the p5.js library in JavaScript. The visualization is presented as a web page, which can be executed either in WebStorm or using a simple Python server.

The visualization process involves two CSV files as input. The first file captures the relationships between characters, including the connected character pairs, the type of their link, and the weight of the connection. The link type is represented by a numerical value ranging from -5 to 5, where values close to -5 indicate an antagonistic connection, values around 5 represent an allied connection, and values around 0 signify a neutral connection. The weight indicates the frequency of character interactions and is a value between 0 and 1. The weights of all connections sum up to 1. The second input file contains information on character sentiment, the probability of each character being a protagonist, and descriptive adjectives associated with the character. Sentiment is represented by a numerical value between 0 and 1, with negative values indicating negative sentiment and positive values reflecting positive sentiment. Additionally, there is a importance value between 0 and 1 that represents the importance of each character. The importance for all characters sum up to 1.

The output is a graph that visualizes the relationships between characters. In this graph, each character is represented by a vertex, while the connections between characters are depicted as edges. The thickness of the edges is proportional to the frequency of interaction between characters. Additionally, the edges are color-coded to represent the type of connection. A red color indicates an antagonistic connection ('sovražnik'), gray represents a neutral connection ('nevtralno'), and green signifies an allied connection ('zaveznika'). The brightness of the color further signifies the strength of the connection. The position of the vertices is determined by the sentiment of the characters and their likelihood of being a protagonist. Characters who are more important are being positioned closer to the center and are represented by larger icons. A numerical value displayed above their respective icons represents the

character's sentiment in the story, ranging from 100% (highly positive) to 0% (highly negative). The position of characters is also influenced by their sentiment, with those sharing similar sentiments being located closer to one another. The visualization is interactive, allowing you to hover over the character icon and see a bubble displaying adjectives that describe the character.

Results

Name entity recognition

We evaluated the outputs of the name entity recognition component and from coreference resolution. We focused mainly on which characters were identified (found at leas once in text) and which characters were left out. Therefore we define two metrics: $precision = \frac{TP}{TP+FP} = \frac{charactersthatwerecorrectlyidentified}{charactersthatwereidentified}$ and $recall = \frac{TP}{TP+FN} = \frac{charactersthatwerecorrectlyidentified}{charactersthatshouldbeidentified}$. Precision tells us how many characters that were outputted are actually correct characters from the stories (our visualization should not contain excessive characters) and recall tells us how many of the correct characters were identified (out visualization should not lack characters). We chose those matrices as they are the most important since they clearly show how many FP and FN we have which is the most vital for our visualization. Results can be seen in Table 1. We report both the mean and the variance of precision and recall scores over all stories.

	average precision	precision standard dev.	average recall	recall standard dev.
only NER model	0.5451	0.1303	0.3595	0.0866
only list of characters	0.7083	0.0563	0.8038	0.0904
NER model with list of characters	0.8065	0.0453	0.9597	0.0232
NER model with list of characters filtered	**0.8789**	0.0437	0.6427	0.0874
NER model with updated list of characters	0.6767	0.031	**0.9939**	0.0013
NER model with updated list of characters filtered	0.8062	0.0454	0.6441	0.0932
crr precision	0.8039	0.0470	0.9067	0.0299

Table 1. Evaluation of NER and coreference components.

We can see that if we only used the classla NER model the recall would be very low, which is to be expected as most characters do not have names. The variance of precision and recall is also higher than with other approaches which means that the model is less stable when it comes to predicting different stories. What is surprising is that precision is also quite low for the NER model, which means that the model recognizes too many words as characters. By using only a list of characters generated by ChatGPT we obtain a quite good results both in terms of precision and recall. When the approaches are combined we obtain a quite good results with approximately 0.8 precision and 0.95 recall. As we examined the data we observed that many words that were wrongly detected as characters appeared only a few times (once or twice). Therefore we implemented filtering which excludes the characters that do not appear in more than 10% of appearances over all characters. This did bring the precision to 0.87 but also lowered the recall to 0.64 as some side characters actually do appear very few times. We also tried to increase the recall by updating the list of allowed characters, however this also drastically decreased the precision as some characters (that are important in some tales eg. ovca, konj...) are not even side characters in other stories but they do appear as words.

In our final implementation of the workflow we use the NER model with the basic list of words (not filtered) as it produces the best trade off between precision and recall.

Correference

We applied coreference resolution on the NER model with list of characters which slightly decreased the recall and precision as we can see on Table 1 (The correference component was applied on the NER model with the basic list of characters). Since we can observe a change in the recall this must mean that the component joined some of the characters (since the model predicted the connection between them) that were actually not supposed to be joined.

From our manual analysis we concluded that the connections between the the correferenced elements are usually meaningless and do not reference the same entity at all. For example in "Vrnil sem se, dokler toliko ne zrasem, da bom goden za gosjega pastirja" it added "sem" to the entity of pastir. Some of them are also correct like in "je nekaj zatulilo" it found entity volk in the "je". However since the precision and recall decreased after this component is applied we do not include the correference component into the final workflow.

Character sentiment

In Table 2 we can observe some properties of the data, prepared for character sentiment model. We can see that the majority of characters are positive as is to be expected from fairy tales. On average each story contains around 5 characters but the number deviates by quite a lot, which is also reasonable as some stories contain only the protagonist with very few side characters, while some contain many side characters that appear only a few times.

To evaluate the bert model we split the data into training testing and validation set with 8:1:1 split. To ensure that the class distribution in the test set is the same as in the original set we use stratified sampling. For the metric we use accuracy as on the final graph we are only interested in how many characters were correctly classified. As we can see on Table 3

character count	339
positive character count	205 (0.605%)
negative character count	134 (0.395%)
stories count	70
average character per story count	5.21
character per story count variance	2.28

Table 2. Data statistics.

the models performed the same as the baseline. The models always learned to predict the majority class. The reason for this is probably too small dataset to train the model.

context of the entity	Accuracy
baseline (majority class)	0.6176
5 words before, 5 words after	0.6176
10 words before, 3 words after	0.6176
3 words before, 10 words after	0.6176
contained sentence	0.5
6 words before, 6 words after, no entity word	0.6470
contained sentences separated	0.3271 (baseline 0.6729)
6 words before, 6 words after, no entity word separated	0.6729 (baseline 0.6729)

Table 3. Results for the fine-tuned BERT model

We then used the same datasets on a custom model. As we can observe the model in this case performed better than the baseline. The best performing context was simply including the whole sentence of the occurrence. After some testing we discovered that the model focused largely on the name of the character in question. For example ”grda hudobna čarovnica” and ”lepa prijazna čarovnica” were both classified as negative. This could be a problem when classifying characters such as ”Jaka” (which is neither positive or negative by itself). Therefore we also constructed a dataset where we only included 6 word before and after occurrence of the character. However as we can see this did not improve the results.

We further set the classification task in a different way. Instead of classifying all occurrences at once we classify each occurrence separately. The train test split is the same as in the previous setting (same characters), however the number of samples is increased and therefore the baseline is also slightly changed (it increased to 0.6729 which makes sense as the good characters are usually protagonists and therefore have more occurrences). However the models performed worse on this task.

Based on the discussed results the best performing model was the custom model trained on whole sentences which we then retrained on the whole dataset and used in our workflow.

context of the entity	Accuracy
baseline (majority class)	0.6176
5 words before, 5 words after	0.6764
10 words before, 3 words after	0.6176
3 words before, 10 words after	0.6470
contained sentence	0.7352
6 words before, 6 words after, no entity word	0.6764
contained sentences separated	0.7009 (baseline 0.6729)
6 words before, 6 words after, no entity word separated	0.6729 (baseline 0.6729)

Table 4. Results for the custom model

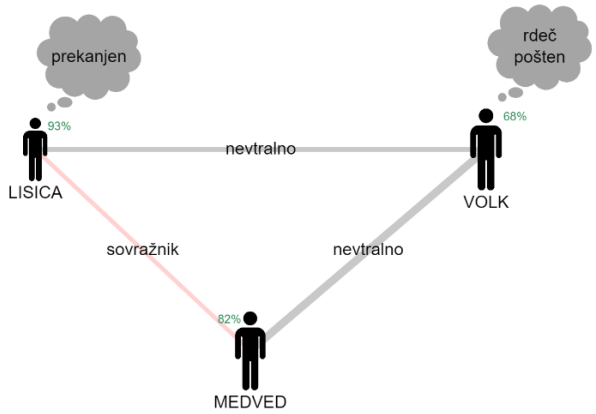

Figure 1. Visualization of characters and their relationships in the story Lisica, volk in medved.

Qualitative analysis of the graph

On Figure 1 we see a good example of the visualisation - a graph that provides a clear and intuitive representation of the characters and their relationships in the story, which can aid in literary analysis and storytelling. In this fable, 'lisica' manipulates 'volk' and 'medved' which explains slightly negative connection although thew were all friends in the beginning. The adjectives 'pretaknjen' and 'pošten' are accurate and highly informative for this case. The only problem with this sample is that lisica perhaps should be recognized as a negative character, however at the beginning of the story it was the victim and those occurrences might have confused the model.

Another quite good example is seen on Figure 2. Connections are true to interactions between the characters, although not all classifications are correct. Each character is assigned true sentiment values and accompanied by well descriptive adjectives. The graph includes all significant characters from the story.

On Figure 3 we can observe an example where the workflow performed very poorly. The main problem with is already at the beginning at the entity recognition. We can clearly see that most of the characters do not actually fit into the story.

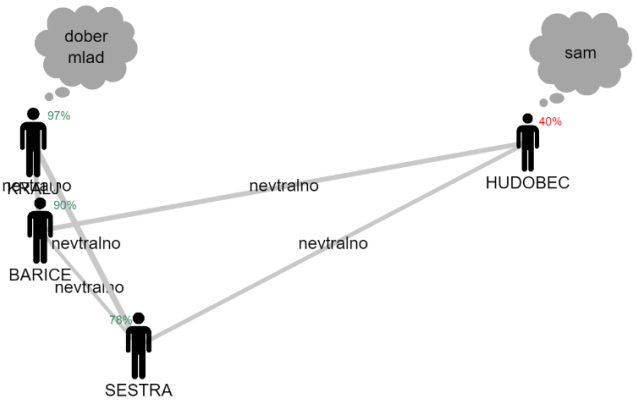

Figure 2. Visualization of characters and their relationships in the story Tri revne deklice.

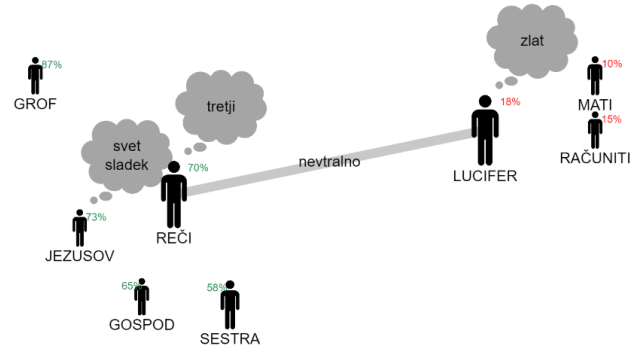

Figure 3. Visualization of characters and their relationships in the story Lucifer se ženi.

Consequently all the other components that use the data from entity recognition component could not perform well as the input data was faulty. However in this case we can still see that the character sentiment model performed very well for the characters that are actually correctly detected.

Discussion

From what we have seen in the previous section we can say that the performance of the component depends highly on the inputted story. Mainly the largest issue is detecting the correct characters. Since in some cases ”gospod” might be an actual character while in other it can only be used as a reference to another character. This problem could be solved with correference model, however it was not included since it performed poorly and only corrupted the result.

The extraction of properties works well on some cases, for example on Figure 1 we get adjective ’prekanjen’ for ’lisica’ and in some other cases we get adjectives for ’zver’ such as ’divji’ and ’požrešen’, for ’pastir’ we get ’zadovoljen’ and ’brezskrben’. On the other hand in one of the stories entity ’babica’ get ’velikanski’ which is not correct, since the story is talking about giant doughnuts. If the extractor finds any adjectives, they are usually correct and meaningful, but there is a portion of stories where none of the properties are extracted. This is not necessarily problematic as some entities are simply not described in such a way. However in other cases the cause of this problem could be the entity recognition component, which does not find all of the entity occurrences and therefore misses the descriptive adjectives. The properties extractor is very simple and it check only the adjectives in the entity neighborhood, which is another part of the mentioned problem as it cannot obtain descriptions that are further away, nor can it confirm that the adjective actually describes the entity.

The performance of the relationship extraction component and the character’s importance component (protagonist detection) are very dependant on the performance of the entity recognition component and we can clearly see them fail on Figure 3.

For the sentiment analysis part we reported in the section above the custom model did improve compared to the baseline (from 0.6176 to 0.7352). After some qualitative analysis of the model outputs we discovered that the model predicts very well very positive and very negative characters (eg. string ”prijazna princesa” has a score very close to 1 and ”hudobna čarovnica” has a score very close to 0), however it performs worse when the context is not so clearly positive or negative. Furthermore in most characters in fairy tales are either positive or negative, however some that do not appear as many times wont necessarily appear in good or bad contexts. Therefore the introduction of neutral class would probably be beneficial.

Conclusion

We have seen that the workflow can produce visualizations that can provide a very intuitive idea about which characters appear in the story and the relations between them. It also provides descriptive adjectives which can help us further understand the characters. However in some cases the workflow completely fails as the entities are wrongly recognized at the very beginning.

To conclude, the results of our workflow depend highly on the inputted story. Or more precisely, it depends on the performance of the entity recognition model on a given story as all the following components of the workflow depend on it’s output. Therefore to further improve our workflow and make it more stable we would have to improve the entity recognition component by fine tuning a NER model on such data or define more precise rules for character detection.

References

[1] Natthawut Kertkeidkachorn and Ryutaro Ichise. T2kg: An end-to-end system for creating knowledge graph from unstructured text. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[2] Igor Melnyk, Pierre Dognin, and Payel Das. Knowledge graph generation from text. *arXiv preprint arXiv:2211.10511*, 2022.

[3] Eric T Nalisnick and Henry S Baird. Character-to-character sentiment analysis in shakespeare's plays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 479–483, 2013.

[4] Lucie Flekova and Iryna Gurevych. Personality profiling of fictional characters using sense-level links between lexical resources. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1805–1816, 2015.

[5] Rene Markovič, Marko Gosak, Matjaž Perc, Marko Marhl, and Vladimir Grubelnik. Applying network theory to fables: complexity in Slovene belles-lettres for different age groups. *Journal of Complex Networks*, 7(1):114–127, 08 2018.

[6] Jože Bučar, Martin Žnidaršič, and Janez Povh. Annotated news corpora and a lexicon for sentiment analysis in slovene. *Language Resources and Evaluation*, 52:895–919, 2018.

[7] Digitalna knjižnica slovenije - dlib.si. https://www.dlib.si/. Accessed: 2023-05-26.

[8] Jože Bučar. Slovene sentiment lexicon JOB 1.0, 2017. Slovenian language resource repository CLARIN.SI.