Maastricht University

Information Retrieval & Text Mining

Inkheart Information Extraction

Abel de Wit
(i6139815)

A project under supervision of Prof. J. Scholtes
May 27, 2021

Abstract

In this research an exploration is made of several information retrieval techniques in order to extract information and answer several questions about the characters and events in the books. First Named Entity Recognition (NER) is used to find the main characters of the book. These named entities are then refined using co-reference resolution. The extracted characters are then used to create a relation map of interactions between characters, overall and dynamically. Lastly dependency graphs are used to find the shortest path between a character and a location to map the topological movements of the characters throughout the books.

1 Introduction

The Inkheart Trilogy, written by Cornelia Funke is a story about a man, Mo, who has the ability to bring books to life, literally. On a night where he is reading a book to his daughter he accidentally pulls three characters from the book into the real world. One downside however, is that for everything that comes out of a book, something else must go in. Because of this condition, his wife and two cats are swapped for the three characters and disappear in the book. Mo tries desperately to get his wife back from the book, but to no avail.

What is interesting about the story of the trilogy is that the set of characters consists of people from the 'real world' and fictional characters in the books, mainly the book *Inkheart*. These characters both explore a world inside the book and outside of it, which creates an interesting topological dynamic.

2 Research questions

The goals of this research are to develop an information retrieval system for the three books in this series which is able to automatically answer questions about the who, and where of the characters. The questions tried to answer in this research are:

- Who are the characters in the books? Can we use named entity recognition to find the different characters, and is it possible to extract certain attributes about the characters? Attributes such as whether they are from inside of a book or from the 'real' world.
- Who do the characters interact with the most during the three books? Is is possible to create a relation mapping between the different characters to see who are the main protagonists, and who can be seen as 'side-kicks'
- Where do the characters go throughout the story. As explained above, the topology2 of the characters changes quite drastically as the characters move in- and outside of fictional worlds. Is the information retrieval system able to differentiate these universe changes and model the movement of the characters throughout the three books?

3 Approach

3.1 Pre-processing

3.1.1 Data loading

The books were converted from EPUB format to TXT format using an online tool. The resulting text files could easily be loaded using Python. A result of the conversion was that each page contained the name of the book (e.g. 'Ink 2 - Inkspell'), using a simple Regular Expression, this could be removed from the books.

3.1.2 Data Structuring

The next step was to divide the book into chapters, this to have a better overview of what happens where in the book, and generally have more structured data. This was also achieved using Regular Expressions. The first two books' chapters consisted of just a number, and because the sentences

are all separate, all that needed to be done was look for a line with solely one number and nothing else ('[0-9]+'). For the third book, the chapters were titled 'CHAPTER 1' etc., so for this the Regular Expression 'CHAPTER [0-9]+' was used.

3.1.3 Data Cleaning

In the first two books, Funke starts each chapter with either a quote or poem of differing length after which the real chapter starts. The authors of these quotes and poems are referenced using the '-' symbol, hence making it easy to discard all lines until the first occurrence of that symbol as the first character in a line.

3.2 Named Entity Extraction

3.2.1 Naive NER

To extract the characters of the book, named entity recognition will be used. First, each book will be tokenized in sentences, and those in words. Then the nltk part of speech tagger will be used to give each word per sentence a POS tag. Finally the nltk ne_chunk module is used to create a nested nltk.tree.Tree object. This object can be iterated over to see which labels are given to a word and what children belong to that word. This allows the recognition of the named entity tag ('NE') and also solves the boundary problem for (most of) our data.

There are still some residual errors from the steps above, such as common stopwords and uncommon named entities. To deal with the stopwords, the list of extracted named entities is compared to the nltk.corpus English stopword list, and the words that co-occur are deleted from our list of named entities. The second problem, the uncommon named entities are words that are technically a named entity, but are not a character in our book. To solve this, the amount of occurrences of each word in the NE-list is counted, and words that have a low count are discarded as a named entity/character that appears so little in the whole book that it does not carry any significant meaning.

3.2.2 Co-references

In order to give a better overview of how often a character occurs, and whom it interacts with, their co-references should be resolved as well. To achieve this, the SpaCy extension package *neuralcoref* is used. It creates co-references for a certain search-window (default: 50 words) and creates a list of lists in which the 'main' character is references, together with a list of all the co-references that belong to that character. This module is however trained on news articles and 'real-world' text data, which creates some difficulties and errors for a fictional book. The *neuralcoref* package does allow to specify a dictionary to improve the co-reference finding, the input would be: `{'Mo': ["man", "father"]}`. To prevent weird co-references to be resolved, a method was created to only resolve co-references for named entities that were found with the approach of Section 3.2.1. Secondly, to evaluate this 'filtered' co-reference resolving, a multi-juror approach to evaluating the results was applied. This evaluation was performed showing a block of text in which a co-reference was found, along with the co-reference and what it would be replaced with. The possible evaluations were '*Good, Bad, Neutral*', neutral when the block of text would not provide enough context on whether the co-reference replacement made sense.

3.3 Interaction Extraction

Because a structured dataset with each sentence that contains a person exists, an intersection method could be applied to measure how often the name (or co-reference) of one character and another are in the same sentence (or within some margin). This should allow for an extraction of which characters interact and how often. This can then be visualized in a relation graph, with stronger edges for stronger relations.

To achieve this, the co-references from Section 3.2.2 are resolved in the text, then each combination of the top twenty characters is tested to see if their names occur together in a sentence. These counts are then combined into an adjacency matrix

that can be used to generate a Graph. This graph is then exported in a format that the program *'Gephi'* [1] can use to visualise the graph. This application allows for displaying of node sizes based on occurrence, edge thickness based on the amount of interactions between two characters, and even dynamically display the interactions over 'time' for each chapter in each book.

3.4 Topological Movement Extraction

The last obstacle is to model the topological movement of the characters throughout time. The time dimension is naturally defined by the progression throughout the trilogy, but with part of speech tagging it might even be possible to extract ordinal pieces (e.g. 'a week later') to define the timeline even better. The other part is the extraction of topological places both in the 'real world' and within the book(s). For this, part of speech tagging will also be needed to identify locations, and from that movement of characters should be possible to extract.

Then using the *networkx* package that was also used to model the interactions, a tree graph is created from the *SpaCy* documents for each chapter. This allows for a search of shortest path between two instances, in our case a name and a location. By iteratively looking for the shortest path to a location for each name, for each chapter, we can model the movements of each character throughout the books. The names and coordinates of the locations are hard-coded according to the x and y locations in *matplotlib*. Then an image of the map can be displayed along with a pointer for the location of that person at their location.

A filter was needed to weed out any shortest path between a character and a location, as they could be speaking about the location or something in that direction. Hence the path from the name to the location needed to contain 'movement' keywords such as *'in', 'to', 'at'*. If the path contained one of these pointers, the character is considered to be at that location.

4 Results

4.1 Character extraction

4.1.1 Naive NER

Using the named entity extraction techniques that are described in Section 3.2, the set of characters per book can be intersected with each other, resulting in a list of characters that appear in each of the three books. The resulting list of characters is:

> 'Meggie', 'Dustfinger', 'Fenoglio', 'Farid', 'Elinor', 'Orpheus', 'Basta', 'Mo', 'Resa', 'Capricorn', 'Roxane', 'Violante', 'Silvertongue', 'Mortola', 'Darius', 'Cosimo', 'Gwin', 'Bluejay', 'Mortimer', 'Piper'

It can be noted that from the first four, three are mentioned in the plot summary on the Wikipedia page [2]:

"In Inkheart, the twelve-year-old, ***Meggie****, discovers that her father* ***Mo****, a professional bookbinder, has the unusual ability to transfer characters from books into the real world when he reads aloud."*

The names are confirmed to be correct manually by reading multiple sentences connected to that name.

4.1.2 Co-reference resolution

The co-reference resolution performed by the *neuralcoref* package showed to work well, however it sometimes performed too well. It would resolve any noun, such as *'The book'* and would also see words at the beginning of a sentence as noun, hence resolving references to that too. This would give some co-reference clusters with *'Her', 'Your', etc.*. To combat this, only the co-references found for the top twenty names from Section 4.1.1 were kept, and the rest was discarded as it was not needed for the information retrieval. The remaining co-references were then evaluated manually by myself and one other juror, the results were put in a table where the *'neutral'* labels are ignored for the calculation

of the Kappa distance. The evaluation results can be seen in Table 1

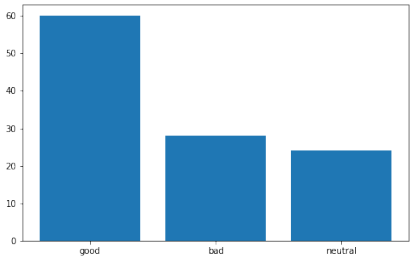

Figure 1: Evaluation of juror A

A/B	Good	Bad
Good	40	20
Bad	7	14

Table 1: Agreement matrix of juror A (myself) and juror B

Then the observed proportionate agreement is:

$$p_o = \frac{a+d}{a+b+c+d} = \frac{40+14}{40+20+7+14} = 0.675 \tag{1}$$

The expected probability that both would say the co-reference is good:

$$p_{Yes} = \frac{a+b}{a+b+c+d} \times \frac{a+c}{a+b+c+d} = \frac{60}{81} \times \frac{47}{81} \approx 0.430 \tag{2}$$

and similarly the expected probability that both would say the co-reference is bad:

$$p_{No} = \frac{c+d}{a+b+c+d} \times \frac{b+d}{a+b+c+d} = \frac{21}{81} \times \frac{34}{81} \approx 0.112 \tag{3}$$

The overall random agreement probability is then:

$$p_e = p_{Yes} + p_{No} = 0.430 + 0.112 = 0.542 \tag{4}$$

So now the Kappa formula will give us:

$$k = \frac{p_o - p_e}{1 - p_e} = \frac{0.675 - 0.542}{1 - 0.542} \approx 0.290 \tag{5}$$

Hence, the inter-rater reliability is can, according to [3] be categorized as 'fair'. The kappa distance is however subject to many external factors such as juror bias.

4.1.3 Visualisations

Now that the character names are available, it is interesting to get a count of how often each name occurs in a chapter, hence showing which characters play a big role where in the books. A count of sub-string occurrences in a piece of text is easily performed, and this is then used to create a stacked plot of the number of occurrences for each character. The result of this process can be seen in Figure [6-8]. These visualisations were done for both the co-reference unresolved and resolved texts, which, as expected, show differences. One of the most apparent differences can be found in the first chapters of book 1 (Figure 6 & Figure 9]) where 'Mo' or 'Mortimer' is very often referenced as 'her father' to 'Meggie'. Hence, in the unresolved occurrence plot, we see that 'Mo' does not occur very often, while after the co-reference replacement, we see that he and Meggie cover most of the introductory chapters of book 1.

4.2 Character interaction mapping

Using the adjacency matrix that was created by looking for combinations of the character names, we can create both a heat-map of this adjacency matrix, and use *Gephi* to visualize the characters as nodes and the amount of interactions as weighted edges.

In the heat-maps in Figure 2, and 3, we can already see which characters interact the most. The main characters, 'Mo' and 'Meggie' have high interaction values for most of the other characters, and they have the highest interaction value between the both of them.

This becomes even more apparent when visualising the co-occurrences as a graph where the edge thickness indicates a higher number of interactions between two characters. A global overview of all the interactions in the three books can be seen in

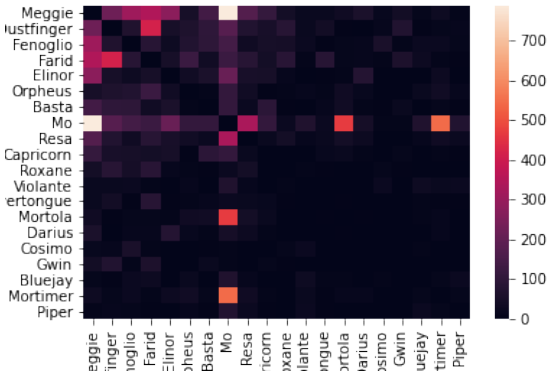

Figure 2: Heatmap of all 20 characters

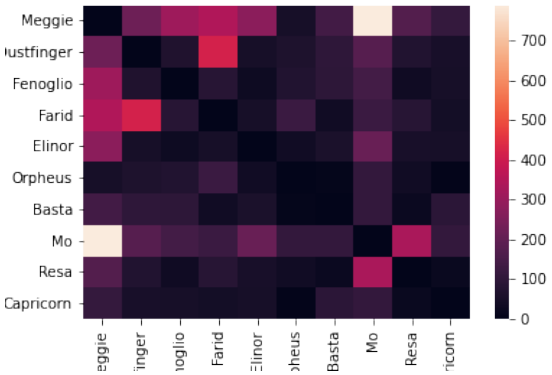

Figure 3: Heatmap of top 10 characters

Figure 4. Here we see again confirmed that Meggie is our lead-character, and she has a strong connection with the following main-characters such as Mo.

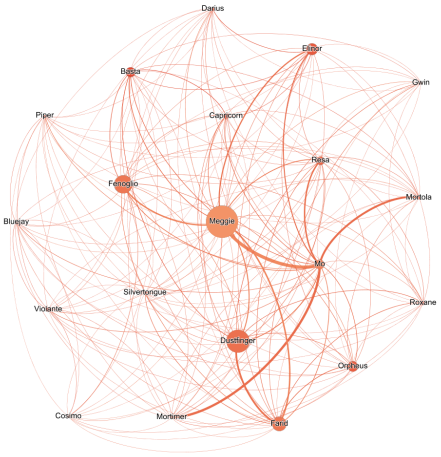

Figure 4: Relation graph for all the books

This graph was also made dynamic, showing the interaction relations per chapter of each book. Because *Gephi* is an application, the dynamic graph has been screen-recorded and can be viewed here as a GIF. The interesting thing about the animation is the fact that we can see certain clusters of characters that are together throughout the story, and between these clusters the characters are (almost) never together, having their own story-lines.

4.3 Topological mapping

Because the Naive NER had trouble with finding location names in the fictional book, it was decided to use a set list of locations that are indicated on the map of *Inkworld*. This list of locations made it possible to find the shortest path for each main character in the books and using '`graphix.shortest_path(source, target)`'. This shortest distance is attempted to be found for each chapter in the dataframe and the resulting list is used to extract the location at some point in the

books for each character.

Such a list for Meggie looks like:

```
[('Ombra', ('book_2', 'ch_8'))
('argenta', ('book_2', 'ch_13'))
('castle', ('book_2', 'ch_20'))
('fortress', ('book_2', 'ch_21'))
('castle', ('book_2', 'ch_25'))
('castle', ('book_2', 'ch_26'))
('castle', ('book_2', 'ch_28'))
('castle', ('book_2', 'ch_29'))]
```

Which can then be used to plot Meggie's location for each location that has been found. The first frame of this animation can be seen in Figure 5

Figure 5: Location of Meggie in Book 2 Chapter 8

The sequence of images has been combined into a GIF for each main character, the folder of these can be found here. While it seems that the Naive NER was not able to extract the locations from the text itself, the shortest distance in the dependency graph did prove to be a good way to find the shortest relation, and hence the highest probability between a character and a location.

5 Conclusion

The results of the experiments and methods that have been tried are very promising. When using the right selection criteria to give bounds to things as co-reference packages, it performs very well and is able to extract the necessary information such as named entities and their co-references. These models are however developed to work on 'real-world' such as existing locations and known person names, in a fictional text these methods obviously perform worse. In future research it might be interesting to see if it is possible to create a classifier that is able to recognize how named entities are used within a sentence to create a better part of speech tag for both fictional character names as well as fictional location names. When this recognition is improved, all other aspects of this research will improve with is, as more accurate location names will result in more accurate dependency graphs, and hence better overview of the topological movement of characters throughout a book.

Nevertheless, this research has shown that text mining and information retrieval from a set of fictional books is certainly possible with the current libraries available in Python. With some manual filtering and insight in the output of a model, good results can be achieved.

All the code and complementary images and GIFs can be found at the following github link: `https://github.com/Abeldewit/IRTM_Inkheart`

References

[1] "Gephi - the open graph viz platform." `https://gephi.org`. Accessed on 2021-05-24.

[2] "Inkheart trilogy." `https://en.wikipedia.org/wiki/Inkheart_trilogy#Plot_summary`, Mar 2021. Accessed on 2021-05-05.

[3] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.

Appendix

Character occurrences without co-references

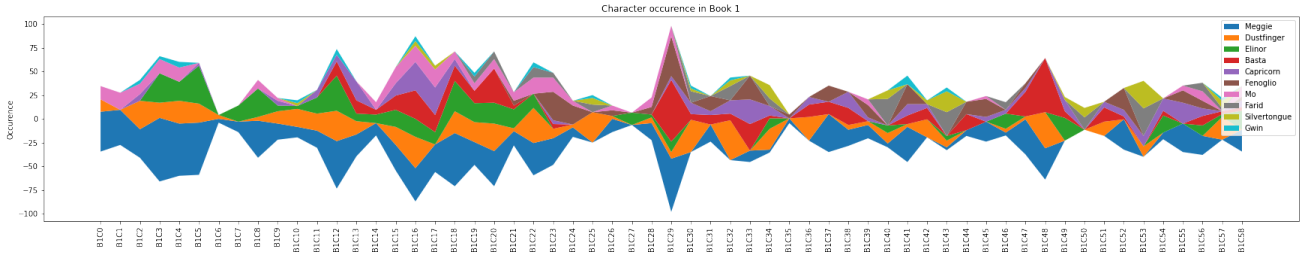

Figure 6: Character occurrences of Book 1 - Inkheart

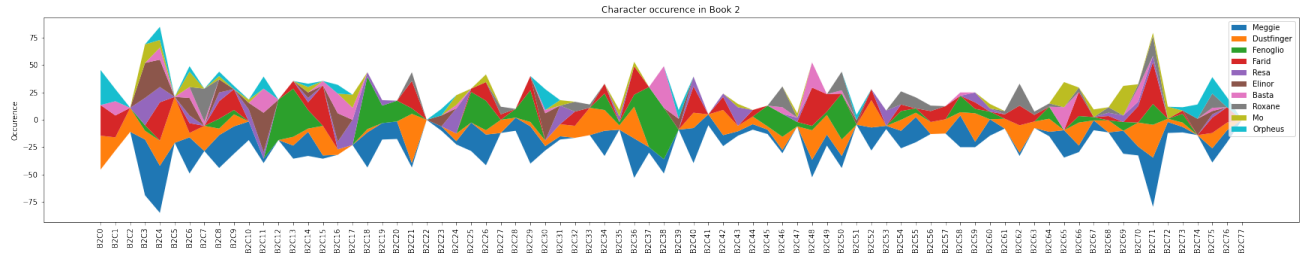

Figure 7: Character occurrences of Book 2 - Inkspell

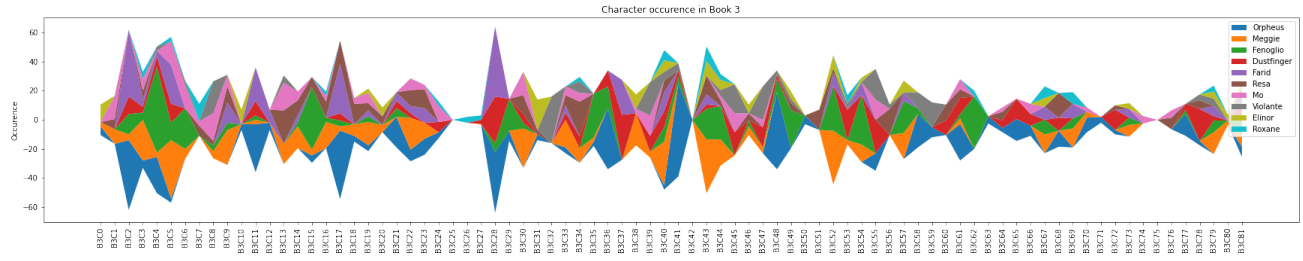

Figure 8: Character occurrences of Book 3 - Inkdeath

Character occurrences with co-references

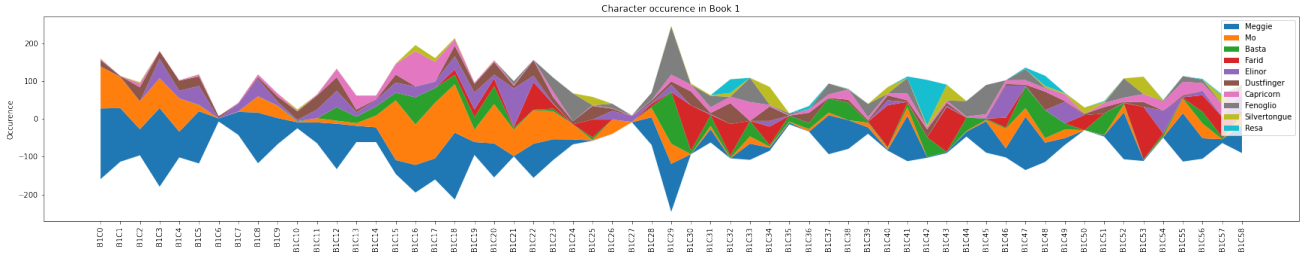

Figure 9: Character occurrences of Book 1 - Inkheart

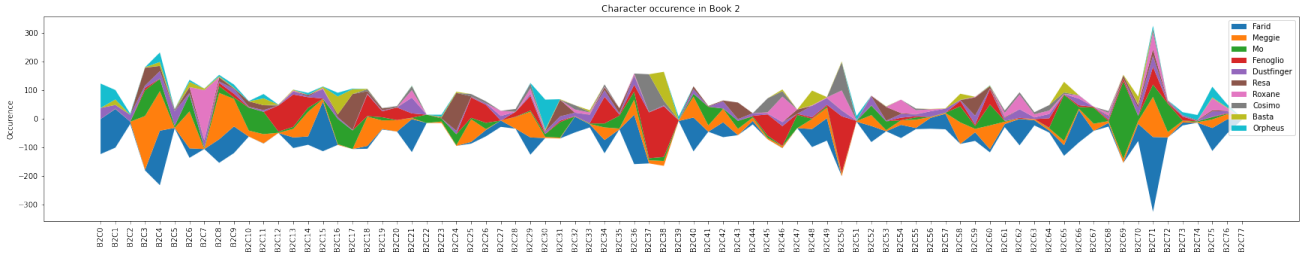

Figure 10: Character occurrences of Book 2 - Inkspell

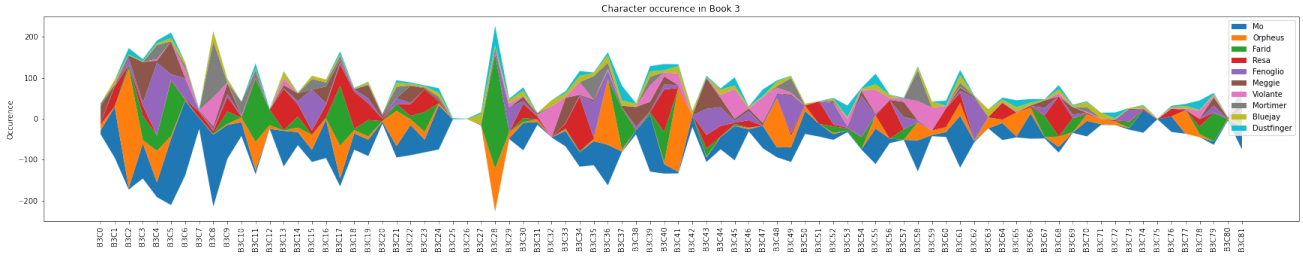

Figure 11: Character occurrences of Book 3 - Inkdeath