Text Mining Homer's Odyssey

Athanasios Rakitzis i6302227

Angeliki Vogiatzoglou i6302869

Project for Information Retrieval and Text Mining

May 2022

J.C. Schoeltes   Thales Costa Bertaglia

Dept. of Knowledge Engineering
Maastricht University

1 Abstract

The original idea for the project was to create a map of Odysseus' travels in the book 'The Odyssey" by ancient Greek epic poet Homer. Afterwards we decided to infer important information about the book's characters and locations alongside with its most prevalent topics. Lastly we decided to find out the sentiment value of each chapter. For entity extraction the open source Flair library was used alongside NLTK. Using the results from NER and by manually gathering coordinates for the locations an animated map of the story was created along with movements of characters between locations. Finally visualizations were created to understand character appearance frequency ,distance covered and other analytical questions.

2 Introduction

The objective of this project was to analyze a classical ancient Greek book, Homer's Odyssey, using state-of-the-art Nature Language Processing algorithms. The research questions that we will try to answer are the following

- Can text analytics be used to infer important information like the main characters,important locations among others?
- Is it possible to produce a map of locations with the aforementioned results to show the movement of characters in the story?
- What is the distribution of the key topics in the Odyssey book?
- What is the sentiment value of each chapter of the Odyssey book?

The above-mentioned questions led the development of the project. The main techniques we used to tackle these questions were Named Entity Recognition, Topic Modeling and Sentiment Analysis.

Delegation of work	
Athanasios Rakitzis	***Angeliki Vogiatzoglou***
Named Entity Recognition with Flair and NLTK	Topic Modeling
Character - Location co-occurence and timeline creation	Sentiment Analysis
Interactive Animated maps with movement patterns of characters with extracted locations and other maps	LDA model to understand the distribution of the key topics in the book and in each chapter
Exploratory analytics and visualizations on extracted entities and tokens	Statistical analysis to derive the most prevalent topic in each chapter

3 Dataset

At first a translation of the book from Gutenberg website was used but later on changed to Robert Fagle's translated version. The main reason behind this is that the latter does not contain any notes to help guide the reader, something that is in almost every edition of Odyssey . Thus, it is much easier to scan through and remove all "extra" content (like a chapter title) with preprocessing. All in all the dataset was very clean with very few notes needed to be removed.

4 Preprocessing

First of all little preprocessing is needed for Named Entity Recognition. That is because by lemmatizing/stemming, removing capital letters(indicator of entity usually), stopwords and even apostrophes(needed for possesives for relation extraction) , semantic context is lost which makes it hard for the NER algorithm to understand what is an entity and what is not .
For text analysis we followed all the steps in this section. For NER , we only removed "extra" content,split the book in sentences and chunks and word tokenized The steps followed in order are the following :

4.1 Tokenization

Tokenization is referred to as the process of splitting a stream of textual data into words,sentences or other meaningful symbols called tokens. A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing.
There are multiple ways to tokenize the English language , either by white space or some other rule which splits the tokens. For example , different tokenizers handle contractions in a different way. Base [1]NLTK tokenizer splits " "didn't " into " did ", " n't " while others like NLTK's tweet tokenizer does not split the contraction into two parts. Flair uses the [2]**segtok tokenizer**.

For text analysis I used the [3]**NLTK RegexpTokenizer**. The command used was " r'\w+' " : "\w "matches any alphanumeric word character (equivalent to [a-zA-Z0-9_], while "+" matches the previous token between one and unlimited times, as many times as possible, giving back as needed. Even though regexp is harder to learn it's value comes from it's flexibility and ability to easily query something intricately specific.
For NER I used **Flair's segtok tokenizer** while sentence tokenizing first ,so that the sentences are splitted, and afterwards work tokenizing.

[1]nltk.word tokenize(https://www.nltk.org/api/nltk.tokenize.html)
[2]segtok package(https://github.com/fnl/segtok)
[3]RegexpTokenizer(https://www.nltk.org/_modules/nltk/tokenize/regexp.html)

4.2 Lowercase

The next step is to lowercase all the characters in the book so we do not have different tokens for the same word but with capital characters. We also do not need to exclude specific cases since no acronyms exist in the book.

4.3 Stopwords Removal

Stopwords are words or characters that do not provide significance in information retrieval. They are typically articles,pronouns, punctuations or words like "while", "however" etc.
At first, NLTK base stopword corpus was used, but later we switched to an elongated [4]list which yielded better results by removing a greater number of unneeded words.

4.4 Part Of Speeech (POS) Tagging

Part of Speech tagging is the process of tagging(categorizing) a sequence words in a text in correspodence with a specific part of speech, depending on the context and the definition of the word. For our project I used NLTK's POS tagging.

4.5 Lemmatization

The goal of lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For example, *run, runs, ran and running* are forms of the same set of words that are related through inflection, with *run* as the lemma. Stemming is a similar process which usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. However, even though stemming is faster computationally, in our limited testings lemmatization proved more useful.
NLTK's WordNet lemmatizer was used which returns the input token to it's lemma if it can be found in Wordnet. The base lemmatizer works on nouns, while one can specify the other 3 POS tags like verbs,adjectives and adverbs. For this project we focused on lemmatizing nouns and verbs.

5 Named Entity Recognition

5.1 NER with NLTK

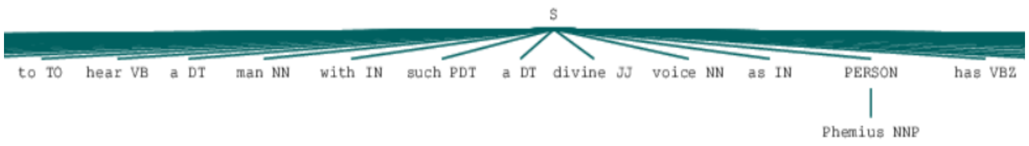
Named entities are definite noun phrases that refer to specific types of individuals, such as organizations, persons, dates, locations and so on. Among the available open-source libraries, we first tried NLTK, using [5]nltk.ne_chunk pre

[4]Stopwords List, fourth comment(https://gist.github.com/sebleier/554280)
[5]nltk.ne_chunk(https://www.nltk.org/api/nltk.chunk.html)

trained classifier. The way it works is, after POS tagging each word in the text, chunking is done. Chunking is the procedure were we segment multi-token sequences and tag them using specific chunk grammar rules which indicate how phrases should be tagged(e.g. noun phrases). These chunk structures can either be represented using chunking trees1 or tags like IOB tags.
The B- prefix before a tag indicates that the tag is the beginning of a chunk, and an I- prefix before a tag indicates that the tag is inside a chunk. The B- tag is used only when a tag is followed by a tag of the same type without O tokens between them. An O tag indicates that a token belongs to no entity / chunk.
Using the command, the next step is to output the tree and traverse it to collect the named entities.

To try it's prediction results , random paragraphs were sampled from the

Figure 1: Example of a chunking tree from the book



book to test the results. By annotating the paragraphs manually , a validation of the predictions was done which resulted in 50% recall and 60% precision. These results motivated us to try different methods to test if we could get better results.

5.2 NER with Flair

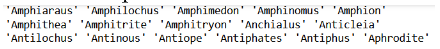
[6]Flair is a powerful NLPlibrary which is open-sourced and developed by Zalando Research. The framework of Flair is built directly on PyTorch which is known as a great deep-learning framework. It uses it's own word embedding system called *contextual string embeddings*, a type of word embeddings based on character-level language modeling. NER and POS-tagging are problems in NLP that can be formulated as sequence labeling problems; that is, text is treated as a sequence of words(which are a sequence of characters) to be labeled by linguistic tags. To handle this procedure, SoTA bidirectional recurrent neural networks (BiLSTMs), and a subsequent conditional random field (CRF) decoding layer are being implemented. First a sentence is input as a character sequence into a pre-trained BiLSTM Language Model, which outputs a these *contextual embeddings*. These are fed into a Sequence Labeler(BiLSTM-CRF) which outputs the tag. This approach produces different embeddings for the same lexical word string in different contexts, and is able to accurately capture the semantics of contextual use together with word semantics itself. In short, it takes context into consideration.

[6]Flair framework(https://github.com/flairNLP/flair)

Other word embeddings are also available in Flair(like ELMO or BERT) but will not be used.

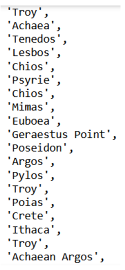
First of all , the book was split into 10 almost equal chunks in order to be proccesable by the computer. Then it was split into sentences, tokenized and tagged. Since the goal is to outline our character's movement patterns, we are interested in extracting Location and Person entities (NER Tagged as LOC and PER respectively). The following tables 2 , 3 are a sample list of extracted locations and characters. The results look convincing, but will later be discussed in detail in the Validation section.

Figure 2: Sample list of Characters extracted



```
'Amphiaraus' 'Amphilochus' 'Amphimedon' 'Amphinomus' 'Amphion'
'Amphithea' 'Amphitrite' 'Amphitryon' 'Anchialus' 'Anticleia'
'Antilochus' 'Antinous' 'Antiope' 'Antiphates' 'Antiphus' 'Aphrodite'
```

Figure 3: Sample list of Locations extracted



```
'Troy',
'Achaea',
'Tenedos',
'Lesbos',
'Chios',
'Psyrie',
'Chios',
'Mimas',
'Euboea',
'Geraestus Point',
'Poseidon',
'Argos',
'Pylos',
'Troy',
'Poias',
'Crete',
'Ithaca',
'Troy',
'Achaean Argos',
```

6 Character and Location co-occurance

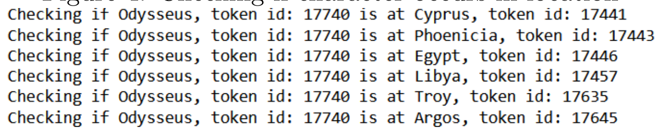
6.1 Approach and results

The approach we followed to check if a character appears in a location to scan nearby sentences whenever we detect a Person entity. More formally, if a character and a location co-occur within a certain period N of words, one can assume that the character is in that location. In this paper by [7]Bonato et al. the idea was implemented for character co-occurance using N=15.
Regarding our project, by setting a token index for every extracted location , we were able to have an indication of time. As the pages and the story progressed, so did the token indexes of the entities. Afterwards, for every entity with the tag 'PER', we scanned if any 'LOC' tag entities appeared **within 40 tokens distance.**(fig. 4) The number was an intuitive guess by using different token distances and checking results. We found 40 token distance to be reasonable distance for a mention of person and location.

[7](https://arxiv.org/abs/1608.00646)

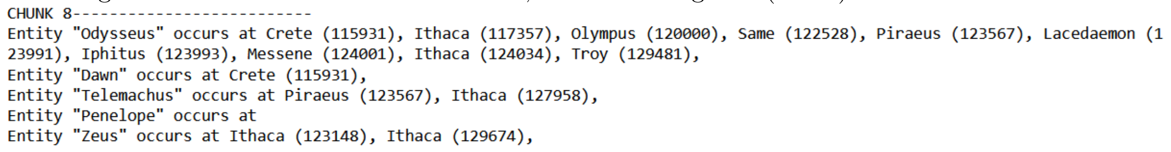
Finally, for every co-occurance, a list(fig. 5) was created which contained character name and the locations he has been to, along with the corresponding index of the location to understand which location our character visited first.

Figure 4: Checking if character occurs in location



```
Checking if Odysseus, token id: 17740 is at Cyprus, token id: 17441
Checking if Odysseus, token id: 17740 is at Phoenicia, token id: 17443
Checking if Odysseus, token id: 17740 is at Egypt, token id: 17446
Checking if Odysseus, token id: 17740 is at Libya, token id: 17457
Checking if Odysseus, token id: 17740 is at Troy, token id: 17635
Checking if Odysseus, token id: 17740 is at Argos, token id: 17645
```

Figure 5: Locations a character visited,with ascending time(index)



```
CHUNK 8-------------------------
Entity "Odysseus" occurs at Crete (115931), Ithaca (117357), Olympus (120000), Same (122528), Piraeus (123567), Lacedaemon (1
23991), Iphitus (123993), Messene (124001), Ithaca (124034), Troy (129481),
Entity "Dawn" occurs at Crete (115931),
Entity "Telemachus" occurs at Piraeus (123567), Ithaca (127958),
Entity "Penelope" occurs at
Entity "Zeus" occurs at Ithaca (123148), Ithaca (129674),
```

6.2 Problems

Some of the issues which emerged were the following.

6.2.1 False entity tags

While the extracted entities on a surface level look correct, there are some issues. One token in particular was very problematic , and that was the city *Troy*. Although a location, it was tagged as a person 34 times, while very few locatiosn were tagged as a person in the entirety of the book. Using the index values , we checked an example sentence(among others) to understand why this occurs.
Sentence: "I am on the trail of my father 's widespread fame , you see , searching the earth to catch some news of great-hearted King Odysseus who , they say , fought with you to demolish Troy some years ago ." → ["Odysseus"/PER, "Troy"/PER]
We believe that there are two reasons behind this.

1. Context is not clear on this specific sentence. Demolish as a verb could indicate both a location and a person.

2. However, while usually context is the most frequent reason, in this case we believe it was also the train dataset of the tagger. The Sequence Tagger of flair was trained on Conll2003[8], a dataset which includes news articles from 12 months. It is possible that Troy appeared as a name many more times than (if any) as a city, and thus, influenced the taggers output.

[8]https://huggingface.co/datasets/conll2003

Another example of a falsely tagged entity is the following :
"No one can say for certain where he died, whether he went down on land at enemy hands or out on the open sea in Amphitrite's breakers."
Here *Amphitrite* was tagged as a location , while she is a person. However it is evident from context that it is not easy even for a human to discern what *Amphitrite* is. The reason it is written like this is because *Amphitrite* in the book is a *God* which has influence over the sea.
All in all, this book is a translation of an ancient Greek epic poem, and thus, it is impossible for a tagger trained from a modern English dataset to work perfectly.

6.2.2 Entity Linking

Another problem to take into account is **Entity Linking**[9] and **Anaphora Resolution**[10]. In general, entity linking is defined as the process of linking named entities found in unstructured texts to records of a knowledge base. For this specific project no such knowledge base exists. While we did not handle it properly, before many visualizations or outputs we manually linked together specific entities that are referring to the same person , like "Pallas" and "Athena", which was the most frequent alias name in the book. Others would include "Zeus" and " Father of the Gods" etc.

6.2.3 Past tense

One last issue with our approach is that whenever the book talks about past events('*flashbacks*'), it is handled as if it is the present time. More specifically, while the location token index is increasing, an event which took place in the past could be mentioned. If there is a person near that location in the text, the algorithm would assume he is in that place in the present . However it does not occur with every character, and mostly happens with a specific city, Troy. Arguably, we could remove Troy as a destination, as it makes sense with the story(no character travels to it at present time) and would improve our output.

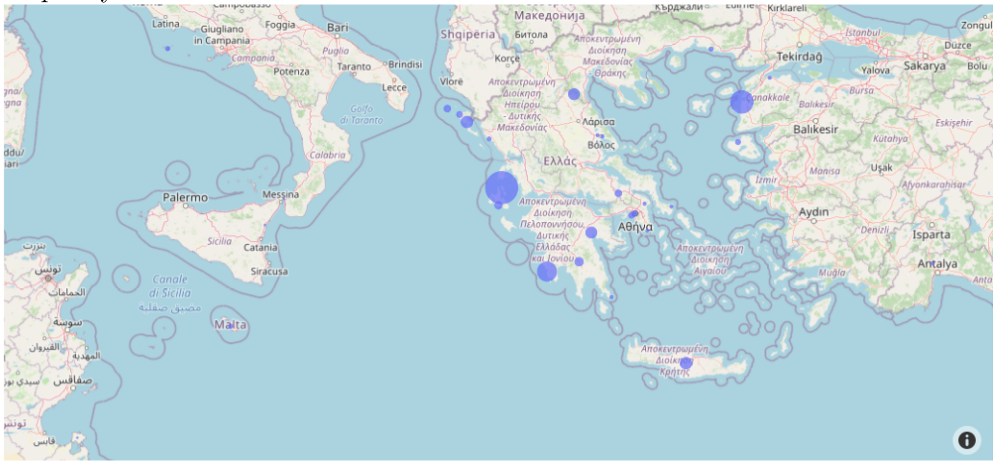
7 Travel Patterns visualized in a Map

The visualization was done using the plotly[11] Library and followed the idea of this article[12] by JP Hwang.
Firstly, coordinates(Longitude and Latitude) were manually collected using Google maps for the 30 most important locations , and then saved in a list. Afterwards, a dataframe containing each extracted person , location(and their corresponding latitude and longitude) and location index was created. Using

[9] https://en.wikipedia.org/wiki/Entity_linking
[10] https://nlp.stanford.edu/courses/cs224n/2003/fp/iqsayed/project_report.pdf
[11] Plotly library(https://plotly.com/)
[12] https://medium.com/swlh/interactive-animated-travel-data-visualizations-mapping-nba-travel-a154a2e7c18d

this information we were able to plot the 30 most important locations in a map, while also showing location frequency with node size. Hovering over the node in the interactive map shows location frequency.

Figure 6: Click here for the interactive version. Node size indicates location frequency.



Following this,

- an interactive map of character travel patterns was created.
- Using location index(time) an **animated map of travels** for every character(in this link, Zeus) was created.

By hovering over the locations, their names can be viewed, and the colours of the lines denote each character movement.

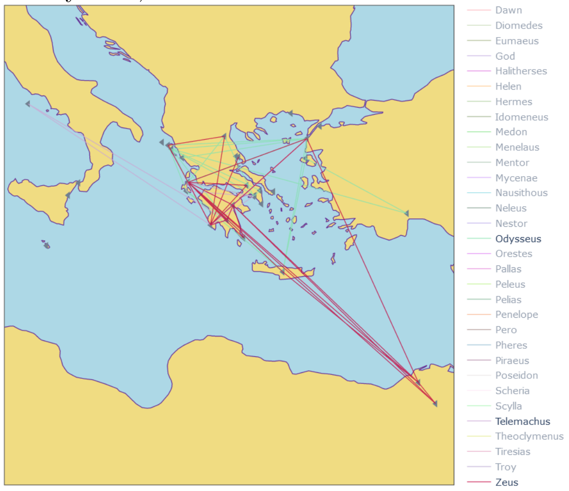
The full list of the interactive animated maps created are being hosted on this website.
Conclusively , our animated map and interactive map work as intented in terms of the entities and locations that were extracted.

8 Validation

The metrics that were used for validating the extracted entities were *Precision, Recall, F-1 score and Kappa distance*. However since the dataset is not annotated , the annotation was done manually after randomly sampling 3 chunks from the book , in total 160 sentences and 3200 words. After the annotation , results were compared with the flair tagger's output on 3 levels

1. Validation on 'LOC' tags only.
2. Validation on 'PER' tags only.

Figure 7: Click here for the interactive version of every character.Here, movement patterns of Odysseus, Zeus and Telemachus.



3. Validation on all tags.

Validation Metrics			
Metrics	'LOC' tags only	'PER' tags only	All tags
Precision	0.58	A0.98	0.72
Recall	0.73	0.84	0.90
F-1 score	0.62	0.90	0.80
Kappa Dist.	0.55	0.67	-

The following calculations were implemented to calcuate these metrics

8.1 Validation for Person 'PER' tags

If the algorithm correctly tagged a person with 'PER' , then we have a true positive. If it wrongly tagged a person as 'PER', then it is a false positive. If it did not detect a person, then it is a false negative. Everything else(tags not having to do with persons) is true negative. Our results were *TP = 70, FP = 1, TN= 22, FN = 13*
Thus , we get :

$$Precision = \frac{70}{70+1} = 0.98$$

$$Recall = \frac{70}{70+13} = 0.84$$

$$F_1 score = \frac{2 * Precision * Recall}{Precision + Recall} = 0.90$$

$$KappaDistance = \frac{2*(tp*tn - fn*fp)}{(tp+fp)*(fp+tn)+(tp+fn)*(fn+tn)} = 0.67$$

As observed, the results for the person entities look promising. Precision means that 98% of the 'PER' tags are true , Recall means that 84% of characters were retrieved/identified, and F_1 score is a combined evaluation of both. Regarding Kappa distance, the agreement/disagreement is between the predicted tags and the actual ones.

8.2 Validation for Person 'LOC' tags

Here, *TP = 11, TN = 83, FP = 8, FN = 4* . The way they were calculated is similar, just for location and not person.

$$Precision = \frac{11}{11+8} = 0.58$$

$$Recall = \frac{11}{11+4} = 0.73$$

$$F_1 score = 0.62$$

$$KappaDistance = 0.55$$

Here we notice a significant decrease in the metrics, especially precision. It is clear that the tagger has issues predicting locations , as almost half of them are not identified correctly.

8.3 Validation for all tags

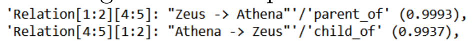
Similarly, calculated as above. In total we get 72% of entities correctly identified and and 90% of total entites were retrieved.

9 Results and Analytics

9.1 Relation Extraction

For this flair's experimental relation extraction tool was used , which was trained over a modified version of TACRED[13] relations. The output of this tool was relations, usually between family members (Figure 9 and 10) . Laertes ,being the father and grandfather of our main characters, is mentioned constantly, alongside Zeus who is mentioned frequently as the father of Gods in the book. For children, symmetrically to the parent relationship, Odysseus,our main character, appears as the most frequent son, followed by Athena,a famous god.

Figure 8: A sample relation



```
'Relation[1:2][4:5]: "Zeus -> Athena"'/'parent_of' (0.9993),
'Relation[4:5][1:2]: "Athena -> Zeus"'/'child_of' (0.9937),
```

[13](https://nlp.stanford.edu/projects/tacred/)

9.2 Main characters and locations

Lastly we wanted to see if our list of entities were actually correctly distributed as we expected. Our assumption is that the main characters would be the Odysseus and his family , followed by the Olympian Gods and everyone else (Figures 11 and 12).
As expected the most frequent character mentioned in the book is the main character, Odysseus.
Regarding the Locations graph, Ithaca and Troy are expected to be the most mentioned locations. One problem here is the falsely tagged "Odysseus" as a Location.

9.3 Distance travelled by characters in km

Lastly, using the latitude and longitude coordinates for each location, we were able to calculate the distance that each character covered thorough the book , using these formulas[14] and calculation tools[15]. Figure 13 in appendix.

10 Conclusions

This section will answer the research questions formulated at the beginning of the project and will also expand on the ones that emerged throughout the course of the project.

Regarding the extraction of entities, the characters were retrieved with fairly perfect ratio. That was not the case however with Locations ,as either many were missed due to anaphora or just falsely classified with different tags. While our recall was not too low, the precision and kappa distance for locations indicate that our results were mediocre.
Regarding the creation of the map, it worked as initially imagined, outlining the trip of every character. The only notable problem here is again ,as mentioned before, the past reference of characters to the city of Troy, which we see many characters returning to time and time and thus creating a false travel pattern for some characters. Could potentially be fixed manually be removing Troy as a destination after a certain amount of pages, as mentioned before.
Finally, this project aimed to give a general insight on Homer's Odyssey by providing analysis and graphs to provide information on the book. Many graphs are located in the Appendix section.
The rest of the research questions are answered in my partner's report.

10.1 Future Research and Potential Improvements

- Anaphora and co-reference resolution are very important for a better and more concrete entity extraction project and should therefore try to be

[14]The Haversine formula(https://en.wikipedia.org/wiki/Haversine_formula)
[15]Calculation toolshttps://www.movable-type.co.uk/scripts/latlong.html)

implemented.

- Proper Entity Linking with a knowledge base should also be implemented for every entity, as mentioned before, as so characters with Alias names/Titles are not ignored.
- A manual annotation of other ancient Greek texts(like the Iliad) could be done in order to train a model to extract entities from this book.
- Lastly character network visualizations would be a nice additional visualization to add to view character relationships.

11 Appendix

Figure 9: Output of relation extraction for parent_of relation

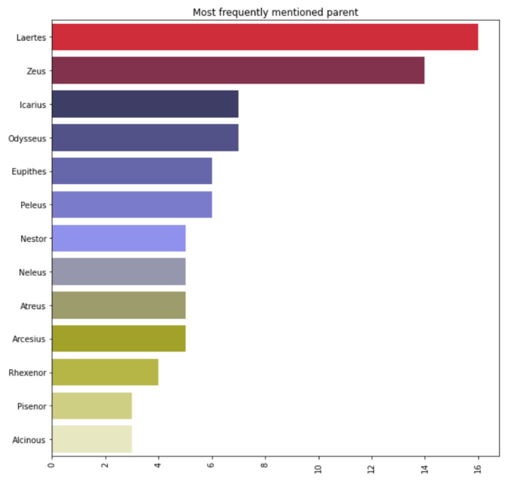


Figure 10: Output of relation extraction for child_of relation

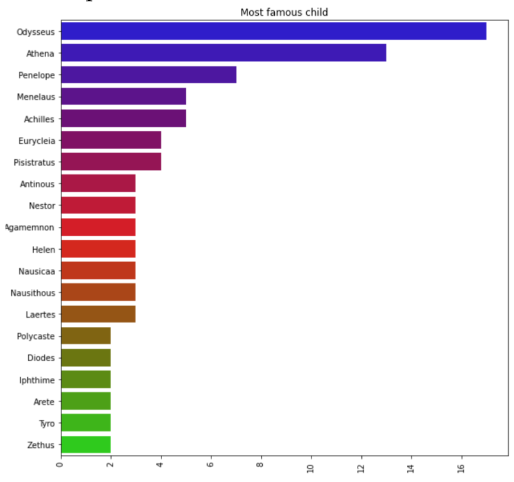


Figure 11: The main characters, by frequency of token appearance

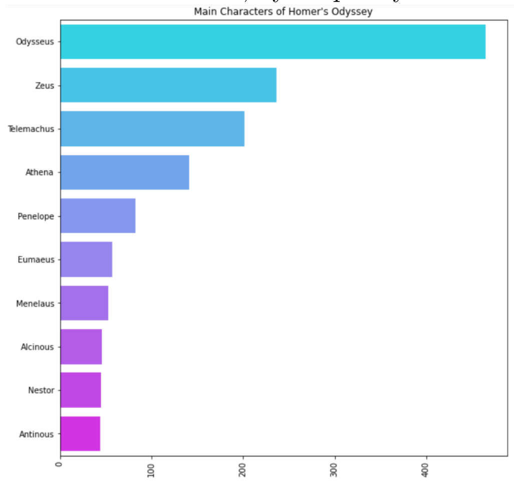


Figure 12: Most important locations, by frequency of appearance

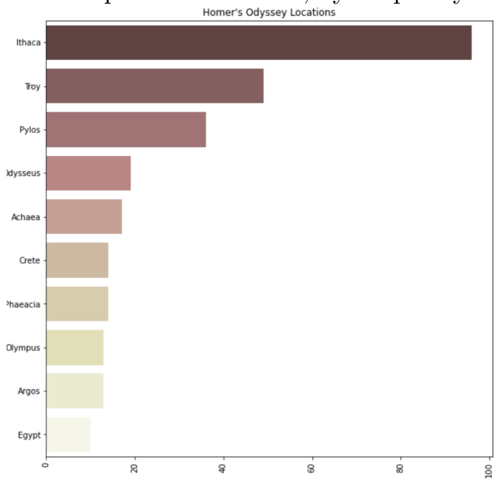


Figure 13: Click here for an interactive version of the plot

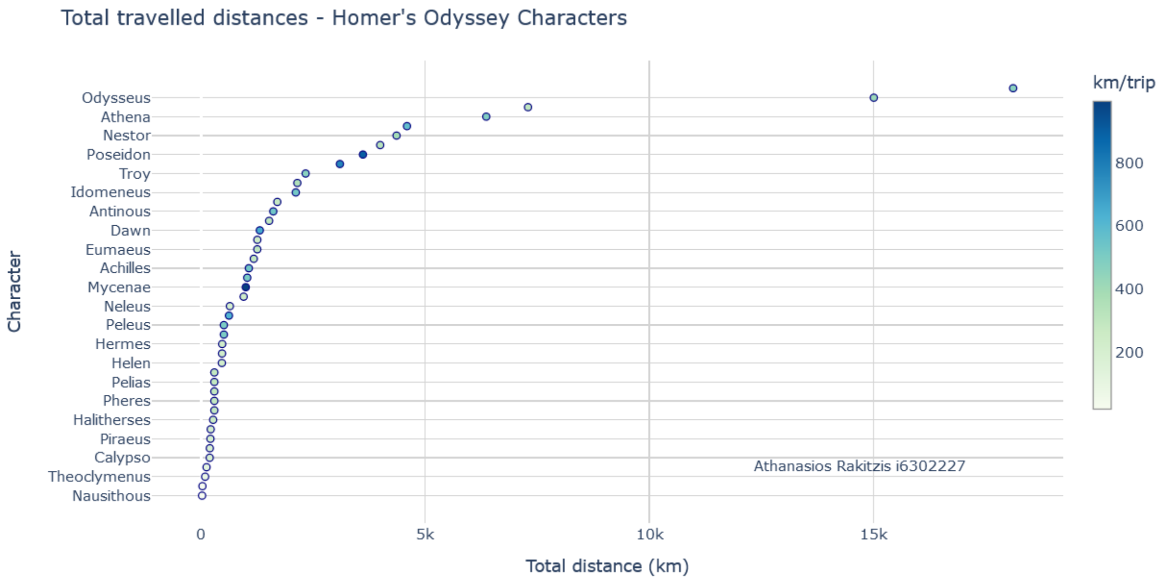


Figure 14: A wordcloud of the most frequent tokens

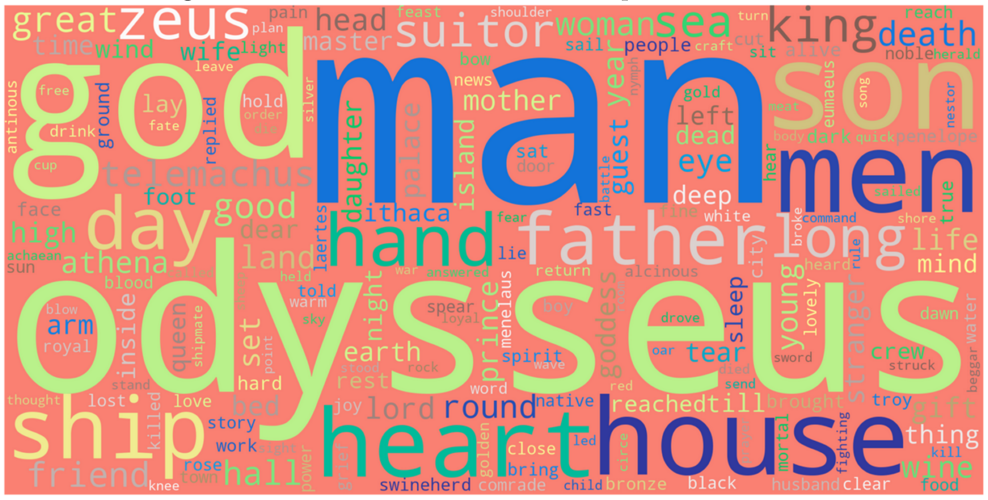


Figure 15: A wordcloud of the most frequent bigram tokens

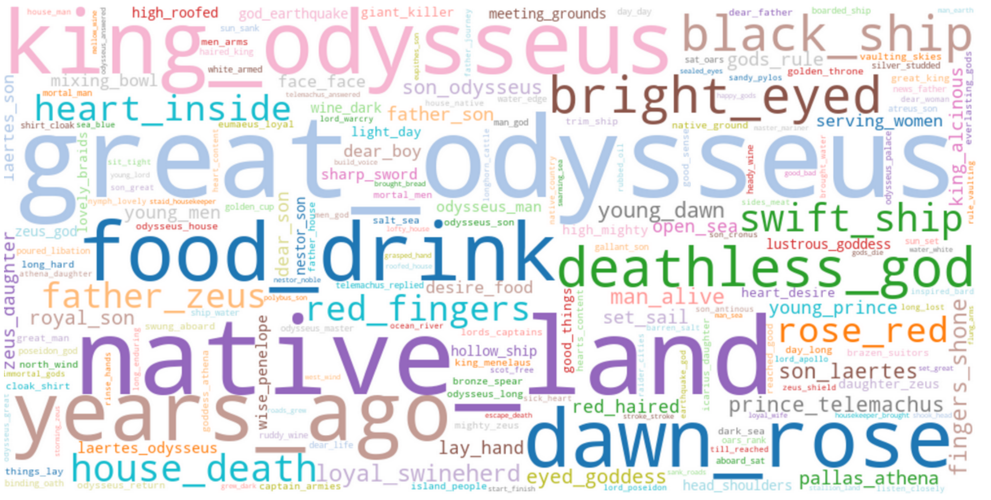
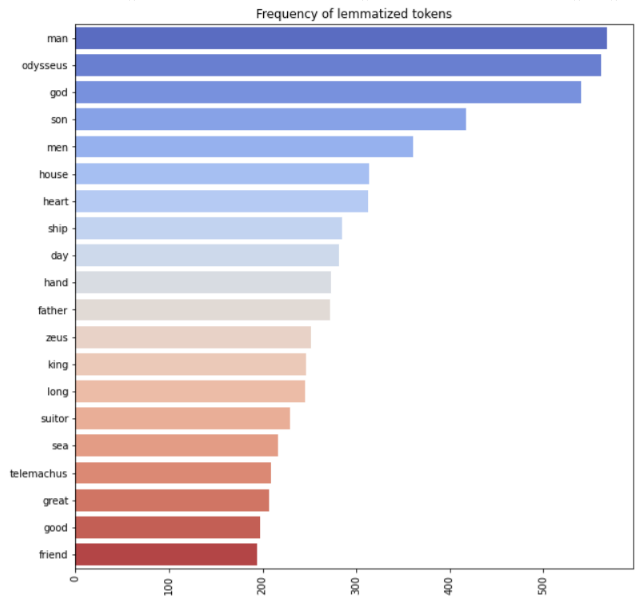


Figure 16: A barplot of the most frequent tokens after preprocessing



References

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., and Vollgraf, R. (2019). Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Bonato, A., D'Angelo, D. R., Elenberg, E. R., Gleich, D. F., and Hou, Y. (2016). Mining and modeling character networks. In *International workshop on algorithms and models for the web-graph*, pages 100–114. Springer.

Akbik et al. (2019) Akbik et al. (2018) Bird et al. (2009) Bonato et al. (2016)