Text Mining - Mahabharata

Balaganesh Mohan, Aleksander Michonski

May 2019

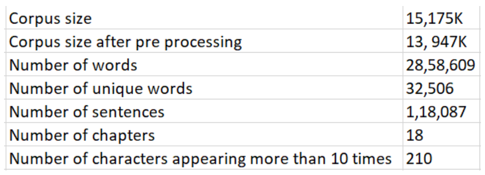
1 Motivation

Topic for this project is to bring Mahabharata closer to the audience without making them read the whole original story which comprises of weighty 18 volumes. After being inspired by previous years presentations, we came up with an idea of making a video of main character travelling around an ancient India area.
But what is Mahabharata about? It is one of the oldest books in the world. We can compare it to Bible or Greek mythology, but it is much older. The story tells about a royal family which fought for a throne. The fight was between the cousins Puravas and Pandavas. Throughout the whole book there are also introduced ancient indian gods who have a supreme power and can help people on earth if we pray enough to them, just like in greek mythology.
Since originally Mahabharata was written in sanskrit and it is easier to operate with english translation where lots of libraries are provided we decided to use english version consisting of original 18 volumes.

2 Dataset

The dataset for this mining project was downloaded from gutenberg[1] open source project for text files. The literature was originally written in Sanskrit which was later translated into English. This version is a translation by Kisari Mohan Ganguli done between 1883-1896. The original size of the text is roughly ten times larger than the Greek epics odyssey and iliad combined.

[1]http://www.gutenberg.org/ebooks/7864



Corpus size	15,175K
Corpus size after pre processing	13, 947K
Number of words	28,58,609
Number of unique words	32,506
Number of sentences	1,18,087
Number of chapters	18
Number of characters appearing more than 10 times	210

Figure 1: Numerical figures of the dataset

3 Methodology

The corpus is narrative heavy, so the text has to be cleaned and noise removal has to be performed before any actual information extraction is done.The method we devised for a comprehensive computational analysis of the Mahabharata epic is as follows:

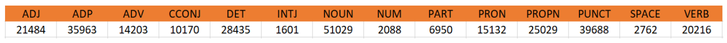
3.1 Pre-processing

For the first part of processing the text we used built in python libraries NLTK and Spacy.

- Removal of supporting texts like authors preface, table of contents, chapter summary, section summary and publisher details.
- Since the whole corpus is in one block of text, it was separated into different chapters(parvas) and then further into different sections.
- Removal of stop words from the pre-established stop word list of the spacy library.

3.2 POS tagging

The pre-processed text was tagged using spacy POS tagger from which different entities of the literature like nouns, proper nouns, adjectives etc.. can be extracted.



ADJ	ADP	ADV	CCONJ	DET	INTJ	NOUN	NUM	PART	PRON	PROPN	PUNCT	SPACE	VERB
21484	35963	14203	10170	28435	1601	51029	2088	6950	15132	25029	39688	2762	20216

Figure 2: POS tagged count for full corpus

3.2.1 Character extraction

- Identify all the proper noun tags from the POS tagged text.
- Identify all the known characters from the Mahabharata arc from the internet and input it as a list.
- Input a thesaurus of characters with aliases to get a total count of all the character mentions.

- Get a count for all proper names in literature which resulted in a total of 426 character which were mentioned at least once in the story.
- Retain only those which are mentioned the most in the story, which will be equivalent to the protagonist of the literature which in our case were Arjuna and Karna each having total mention of 3800 and 2595 times respectively.

3.2.2 Description of the characters

The adjectives related to the extracted proper nouns were analysed to find how the characters were viewed in the story. Some of the main characters were described as follows.

Arjuna is mighty, warrior and fierce
Bhima is mighty, warrior and terrible
Duryodhana is wicked, mighty and desirous
Krishna illustrious, celestial and beautiful

A word cloud visualization will be shown later in the report for the two main protagonists of the story.

3.3 Sentiment Analysis

Identifying the sentiment of a story can explain overall positivity of negativity of the story. But to be more precise we can analyze the sentiment of each volume(parva) and see how the story progresses over the timeline and whether it starts and end in a positive or negative note.

3.3.1 VADER sentiment analysis

VaderSentiment[2] is a library for sentiment analysis which is built over the NLTK framework. This library can be used to analyse the sentiment on word, sentence and document level.
The corpus is split into different volumes and then each volume is stored as a text file. Each volume was tokenized on sentence level and vaderSentiment was used to perform sentiment analysis of every sentence in the volume.

Scoring methodology:
The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive). This is the most useful metric if you want a single unidimensional measure of sentiment for a given sentence. Calling it a 'normalized, weighted composite score' is accurate.
Eg: 'pos': 0.746, 'compound': 0.8316, 'neu': 0.254, 'neg': 0.0

[2]https://github.com/cjhutto/vaderSentiment

The compound scores of each sentences are taken for calculation of the weighted average of each volumes which gives the sentiment value of that particular volume.

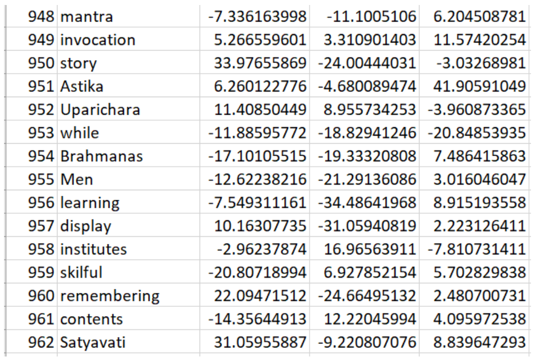
3.4 Similarity extraction

The motivation here is to extract semantic similarity between words in the corpus by converting the word into vectors which will be in higher dimensions, which in turn can be reduced to lower dimension and finally use cosine similarity to analyze semantic similarities, i.e. to answer relationship questions based on the learning.
Skeleton of the approach will be,

- Converting the corpus into sentence tokens and then into a bag of words.
- Apply stopwords function to remove useless words and punctuation's.
- Build model by training word2vec and build a vocabulary.
- The trained word vectors will be in a high dimension, example more than 200 dimension. Using tdistributed stochastic neighbor embedding or t-SNE to reduce this higher dimension to a feasible, analyzable dimension size. Train the above dimensionality reduction algorithms to create a lower dimension dataset. Plot and analyze it for semantics.
- For further analysis and to answer the problem statement, cosine similarity is used to assess similarities between 2 word vectors, to answer similarity questions on the 3rd word vector.

3.4.1 Implementation

- The Paragraphs are split into sentences and in turn this corpus is chopped into words with the help of split() function.
- Stop word and punctuation removal - Improve the dataset by removing the stop words and symbols that does not have meanings. This was done using spacy stopwords function.
- Train word2vec on this processed corpus, thereby converting words into vectors. This is done using Gensim's word2vec library.
- The trained word vectors will be in a high dimension. With the help of t-distributed stochastic neighbor embedding or tSNE reduce this higher dimension to a feasible, analyzable dimension size. T-SNE is implemented using sklearn. The x, y and z coordinates can be extracted for plotting purpose and semantic analysis.



948	mantra	-7.336163998	-11.1005106	6.204508781
949	invocation	5.266559601	3.310901403	11.57420254
950	story	33.97655869	-24.00444031	-3.03268981
951	Astika	6.260122776	-4.680089474	41.90591049
952	Uparichara	11.40850449	8.955734253	-3.960873365
953	while	-11.88595772	-18.82941246	-20.84853935
954	Brahmanas	-17.10105515	-19.33320808	7.486415863
955	Men	-12.62238216	-21.29136086	3.016046047
956	learning	-7.549311161	-34.48641968	8.915193558
957	display	10.16307735	-31.05940819	2.223126411
958	institutes	-2.96237874	16.96563911	-7.810731411
959	skilful	-20.80718994	6.927852154	5.702829838
960	remembering	22.09471512	-24.66495132	2.480700731
961	contents	-14.35644913	12.22045994	4.095972538
962	Satyavati	31.05955887	-9.220807076	8.839647293

Figure 3: Example 3D space word vectors

– As our final goal is to identify relationships between people, extraction of Proper nouns from the above step is a right way to reduce noise.This can be done using spacy POS tagger and proper nouns can be extracted.

4 Results

The outputs and visualizations of all the process done above are shown here. After noun extraction, them most occurring nouns are displayed as a wordcloud.

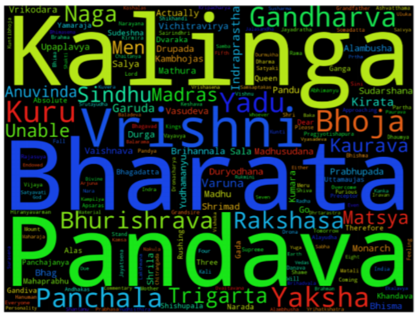


Figure 4: Nouns

4.1 Character extraction

We discussed the methods to extract the protagonists of the story by extracting the most mentioned characters in the literature. The output of it is the character mention count of Arjuna and Karna per volume of the books.

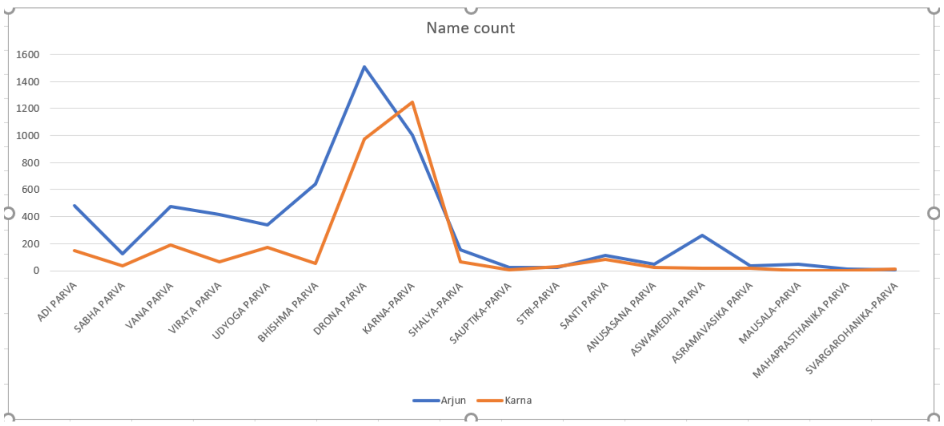
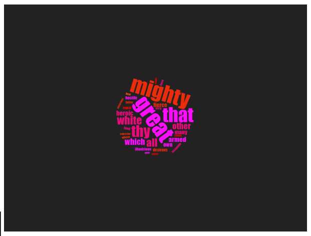


Figure 5: Character mentions per volume

4.2 Adjectives of the protagonist

Next step was to extract adjectives that are used to identify the main characters of the story viz. Arjuna and Karna

[Arjuna]


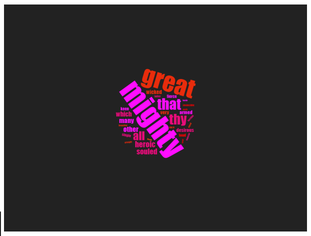
[Karna]


Figure 6: Protagonist Adjectives

4.3 Sentiment Analysis

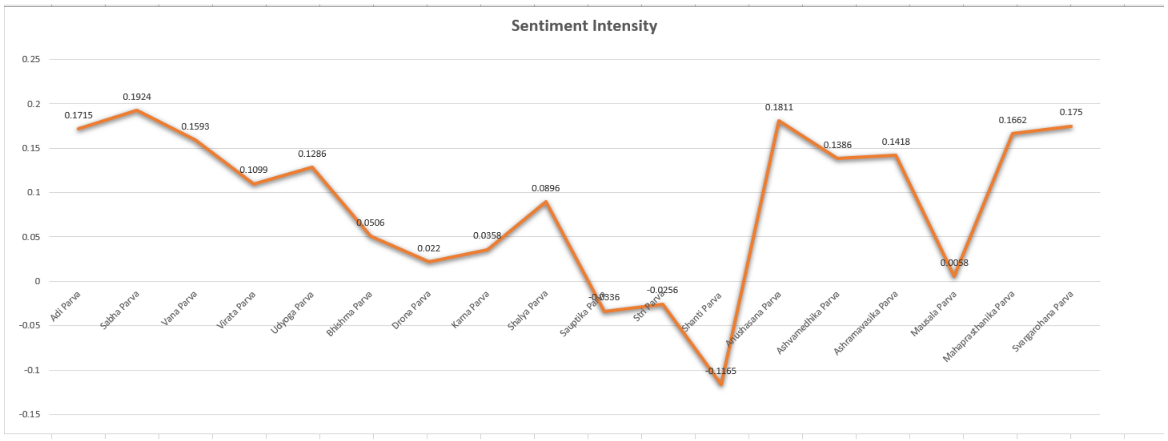
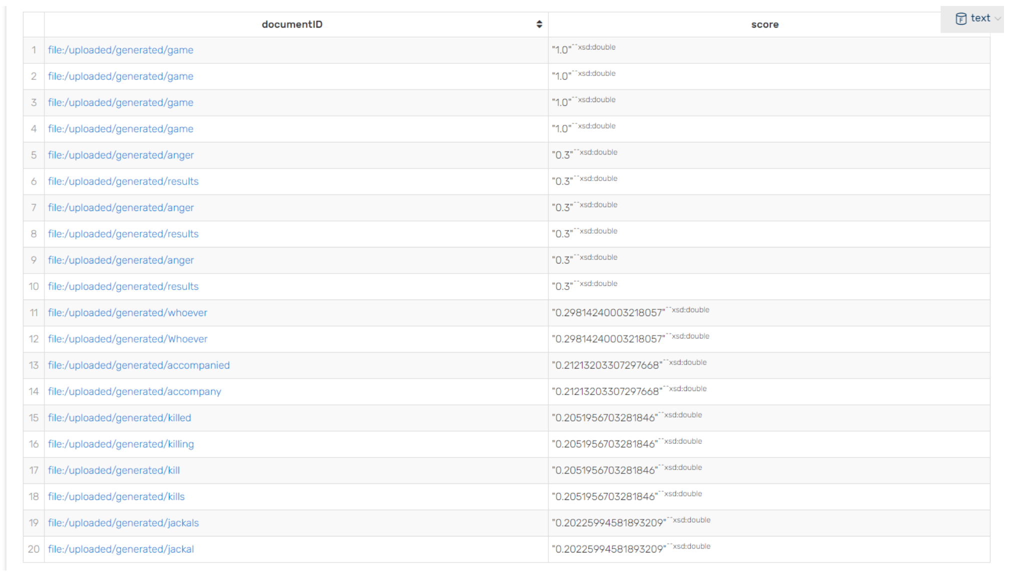


Figure 7: Nouns

4.4 Similarity extraction

In this part we considered several method to present similarity of the words between each other. One of them is 3D vector space decomposition, where similar words are put together composing a circle (fig.9). We achieved it using excel tools. A different one is created from rdf. Created triples were put into graphdb repository where further were processed using there a built in module for finding indexed similarities (fig.8). It allows us type in a chosen word and get back its closest relatives based on calculations of three dimensions x,y,z.



text

	documentID	score
1	file:/uploaded/generated/game	"1.0"^^xsd:double
2	file:/uploaded/generated/game	"1.0"^^xsd:double
3	file:/uploaded/generated/game	"1.0"^^xsd:double
4	file:/uploaded/generated/game	"1.0"^^xsd:double
5	file:/uploaded/generated/anger	"0.3"^^xsd:double
6	file:/uploaded/generated/results	"0.3"^^xsd:double
7	file:/uploaded/generated/anger	"0.3"^^xsd:double
8	file:/uploaded/generated/results	"0.3"^^xsd:double
9	file:/uploaded/generated/anger	"0.3"^^xsd:double
10	file:/uploaded/generated/results	"0.3"^^xsd:double
11	file:/uploaded/generated/whoever	"0.29814240003218057"^^xsd:double
12	file:/uploaded/generated/Whoever	"0.29814240003218057"^^xsd:double
13	file:/uploaded/generated/accompanied	"0.21213203307297668"^^xsd:double
14	file:/uploaded/generated/accompany	"0.21213203307297668"^^xsd:double
15	file:/uploaded/generated/killed	"0.2051956703281846"^^xsd:double
16	file:/uploaded/generated/killing	"0.2051956703281846"^^xsd:double
17	file:/uploaded/generated/kill	"0.2051956703281846"^^xsd:double
18	file:/uploaded/generated/kills	"0.2051956703281846"^^xsd:double
19	file:/uploaded/generated/jackals	"0.20225994581893209"^^xsd:double
20	file:/uploaded/generated/jackal	"0.20225994581893209"^^xsd:double

Figure 8: Similarity index from graphdb

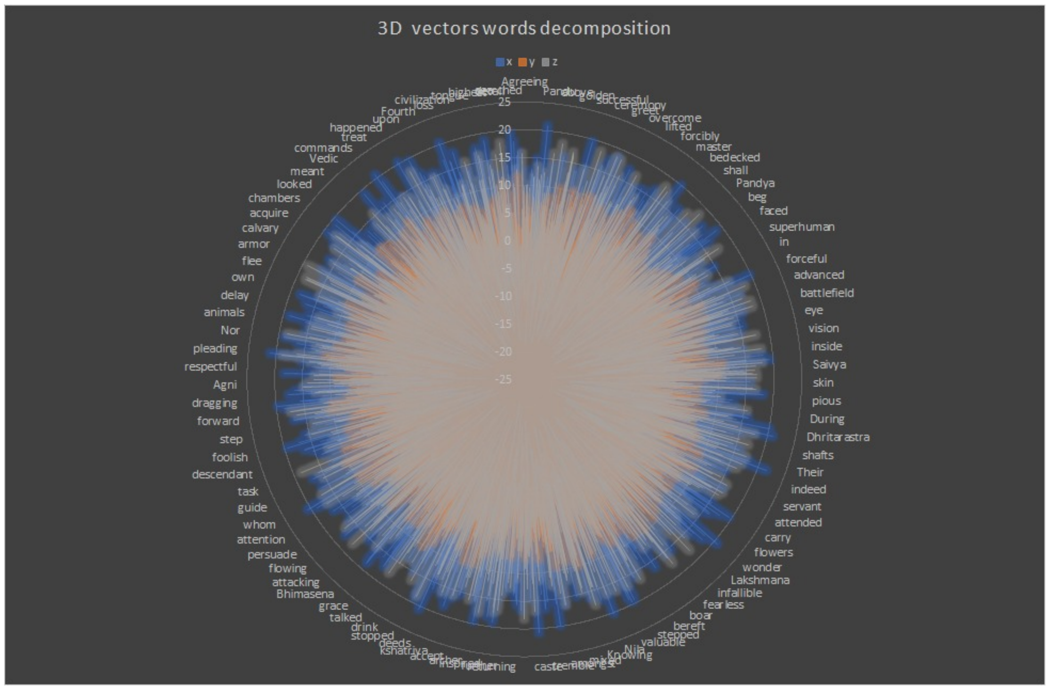


Figure 9: Similarity index from graphdb

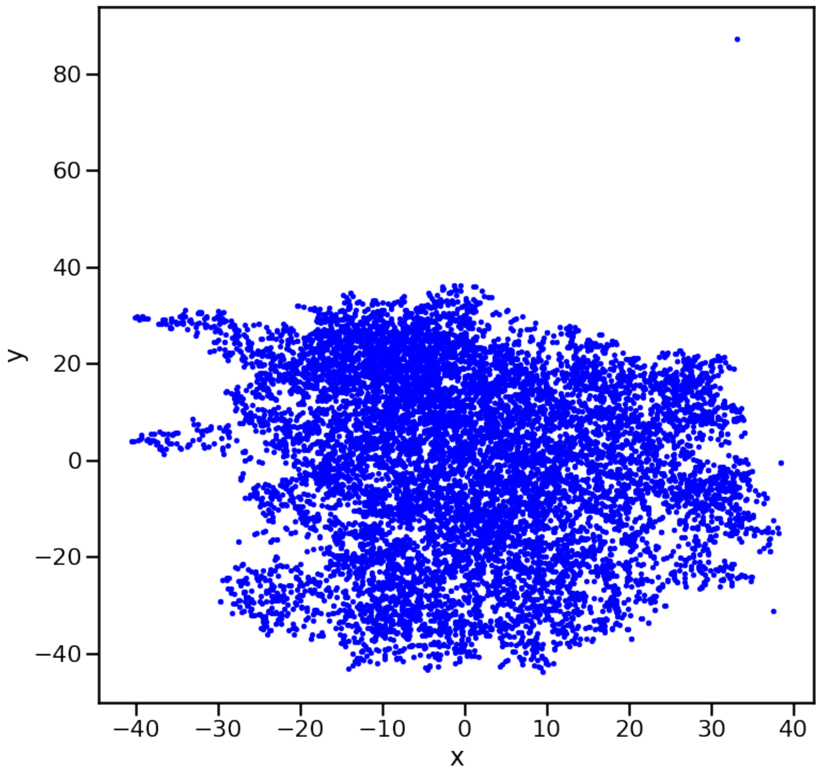


Figure 10: T-sne Scatter plot for full corpus

However, this figure does not help to extract much information because all the unworkable tokens represented, but we can extract meaningful information like closeness of nouns from this by querying a particular tag from POS tagged words. The last figure (fig.10) is the outcome of python plots. As we can see, data can be represented differently using varied tools dependently which environment we feel the most comfortable in.

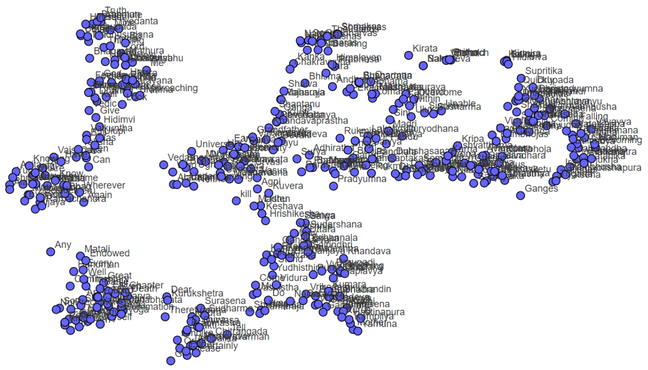


Figure 11: T-sne Scatter plot for nouns in the corpus

This above figure shows only the nouns of the word embedding and it also visualises separate factions or clusters of words which forms the different families in the literature and smaller ones are places and names of divinities or gods.

5 Arjuna's wanderings

This is a part of who, where and when analysis. In the included video it is shown who was the main protagonist, one of Pandavas, journey from happy days in his fathers kingdom to exile, his single pilgrimage to holy places to ask gods for astras (enchanted supernatural powers) while preparing for a war and reunion with brothers. In overall it took him more than 20 years. Map and animation was prepared in qgis. Map was created from manually put points on the background map of current India. Then using Time Manager plugin csv format data of Ajruna's travel was uploaded. As a result we got saved in .png images representing single timeframe. Nextly, Kapwing software was used to put it together into a gif. A preview image of the video has been included below (fig.12).

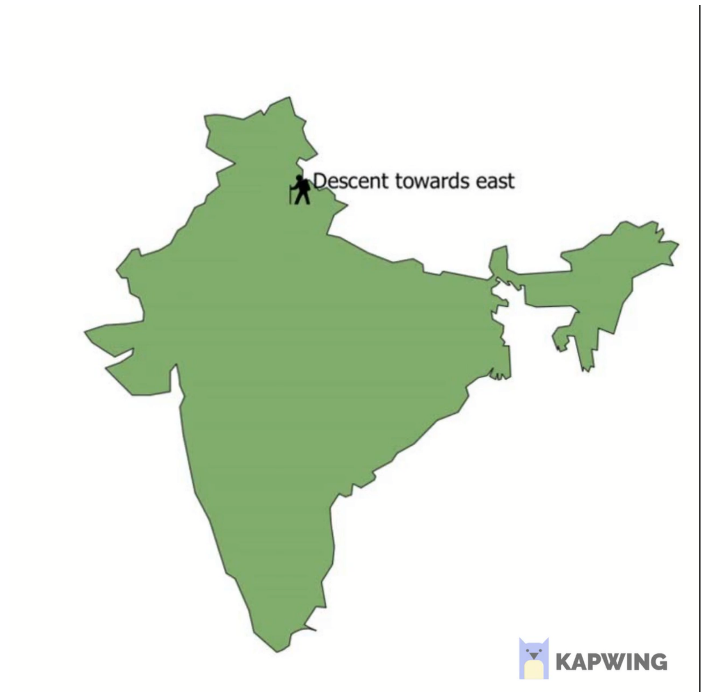


Figure 12: An example frame from Ajruna's wandering's video

6 Conclusion

As we can concurred from the results, there were many meaningful information that was extracted from the literature and there are many more approaches to extract more intricate information from the books.The modern NLP libraries like spacy, vader and unsupervised neural network like word2vec proved to be powerful tools. However, we are aware of several constraints like variability of writing style which can prove our current algorithm to be vulnerable and inefficient for other books. We can use more modern supervised learning methods like BERT to greatly increase the accuracy of the predictions. For future extensions there are many more things we can apply to our idea such as resolving co-references and explore the relations between the main characters.