# **NLP Project**

Part-of-Speech Tagging

Submitted to: **Dr. Sakthi Balan**

Submitted by: **Fondation Seekers**

Group Members:

Shivesh Kaundinya - 19UCS154

Keshav Maheshwari - 19UC060

Roshan Kumar - 19UCS178

ACKNOWLEDGEMENT

We are grateful to our respected teacher, Dr. Sakthi Balan, whose provided knowledge in classes benefited us in completing this project successfully. Thank you so much for your continuous support and presence whenever needed.

Last but not the least, we would like to thank everyone who is involved in the project directly or indirectly.

And we are always thankful to The Great Almighty for always having a blessing on us.

# 

# 

# 

# OBJECTIVE

In this project, our aim is to apply parts-of-speech tagging on two texts. For this purpose, we will first have to apply preprocessing on the texts such as removing running words & stop words, minimising all upper case letters, tokenising all the words, etc. We will also be doing analysis and visualization on the texts to get a better understanding and draw inferences.

The texts we will be using are [The Kingmakers, by Burton E. Stevenson](https://gutenberg.org/ebooks/66728) and [The Return of Sherlock Holmes, by Sir Arthur Conan Doyle](https://gutenberg.org/ebooks/108).

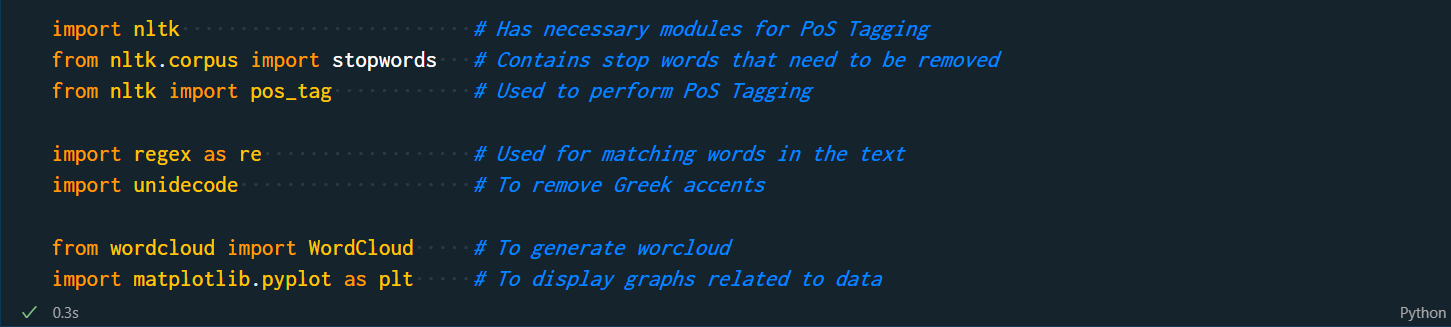
The entire code can be found here: <https://github.com/gmmkmtgk/Foundation_Seekers>

# SETUP

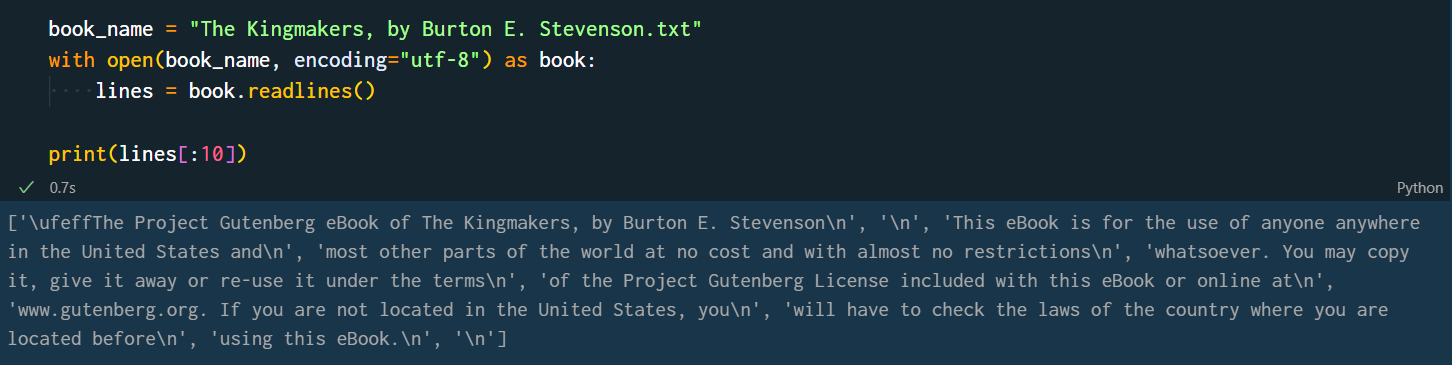
## Importing Necessary Libraries

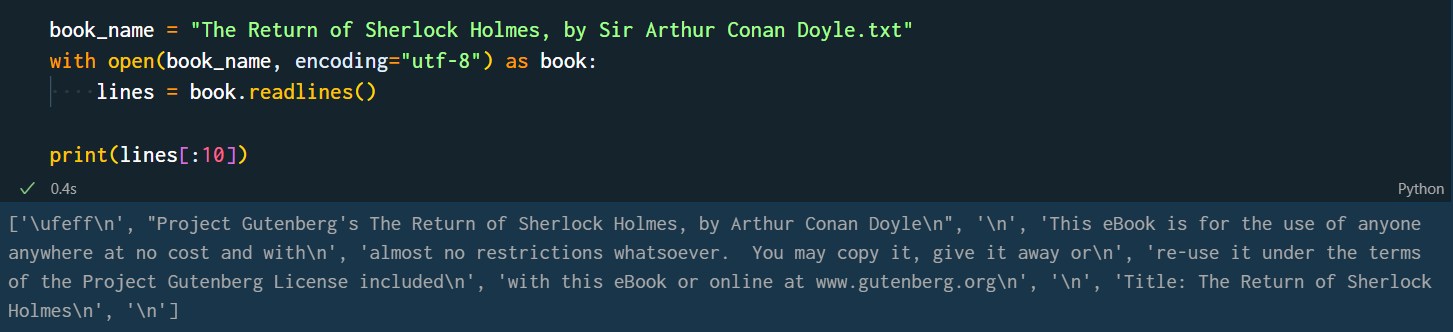
We need to import a few important libraries:

* nltk: This will be used for tokening and performing tagging
* regex: This is required for matching texts that need to be processed
* wordcloud: This is required to create a word cloud for better visualization
* matplotlib: This will be required for displaying charts



## Reading the Texts

Reading a text file is quite trivial in python. We will open the text file and read it line-by-line storing it as a list of strings.



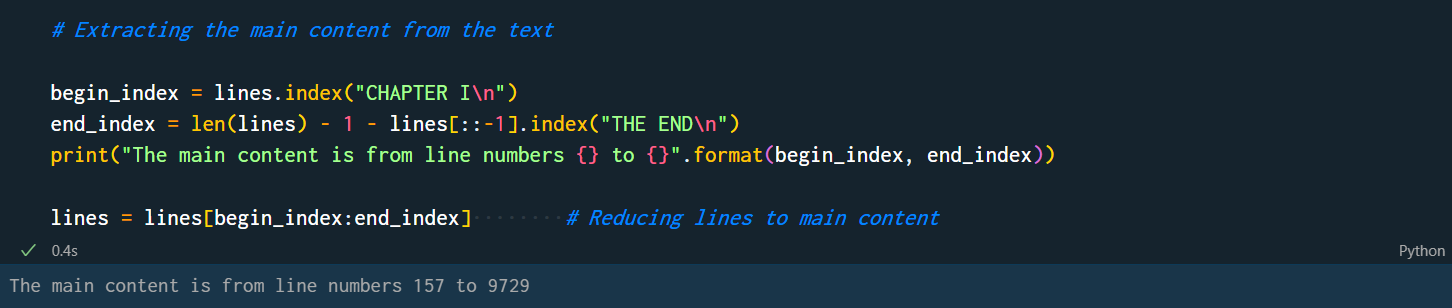
The first and the second texts contain 73693 words and 111974 words, respectively.

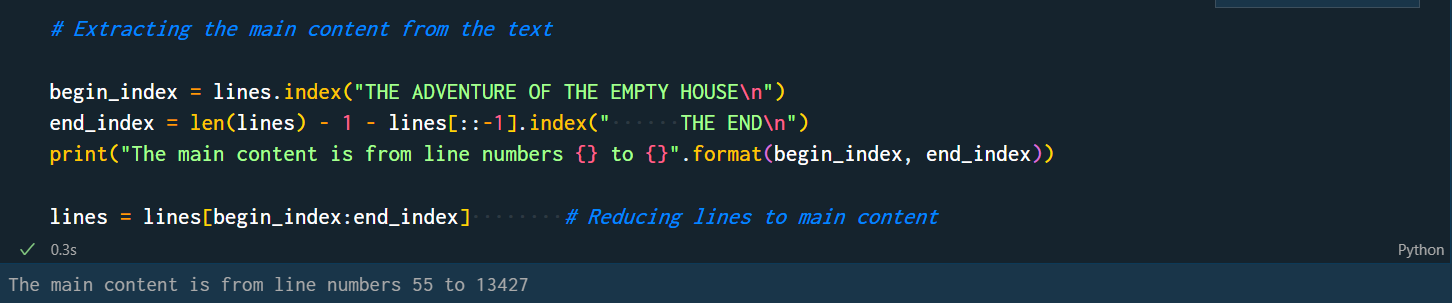
# PREPROCESSING

After a basic setup, it is time to process the raw data and tokenize it.

## Extracting Main Content

For extracting the main content from the text, we will find the index of the first chapter and the index of the last line. We can then reduce the array to strings to just the sub-array between the derived indices.

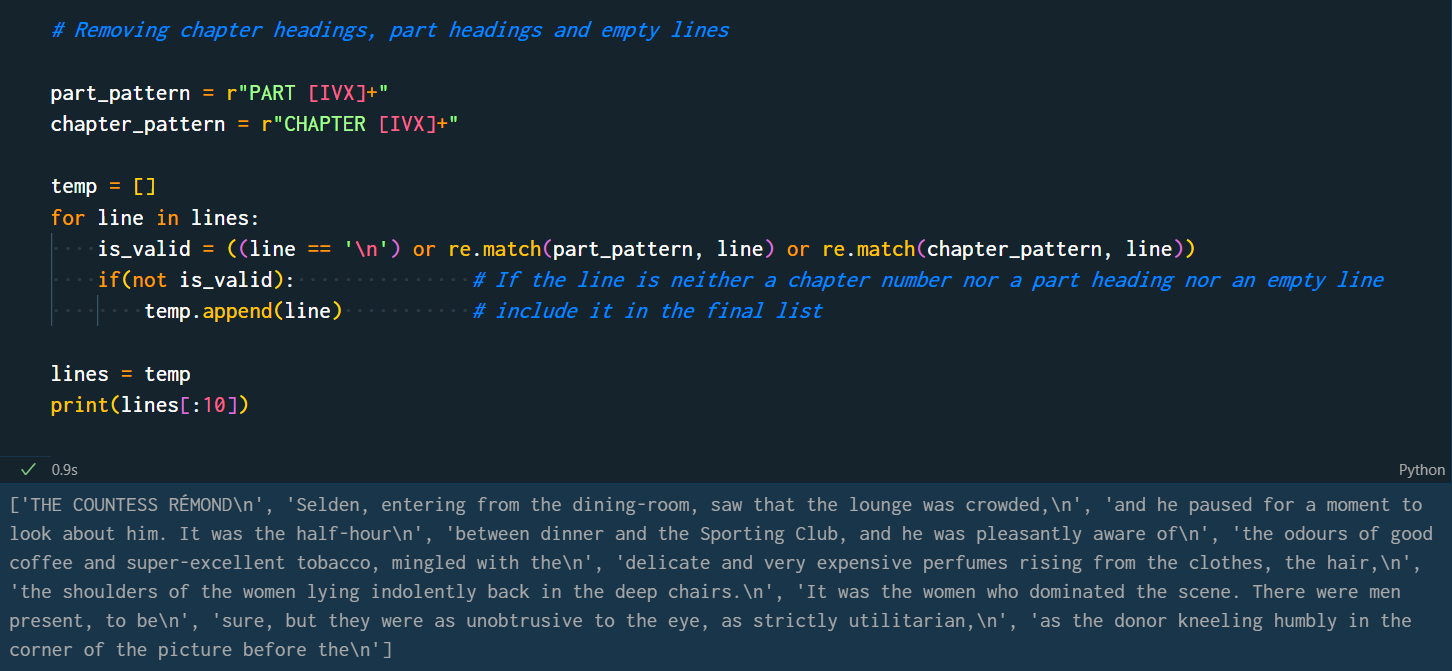


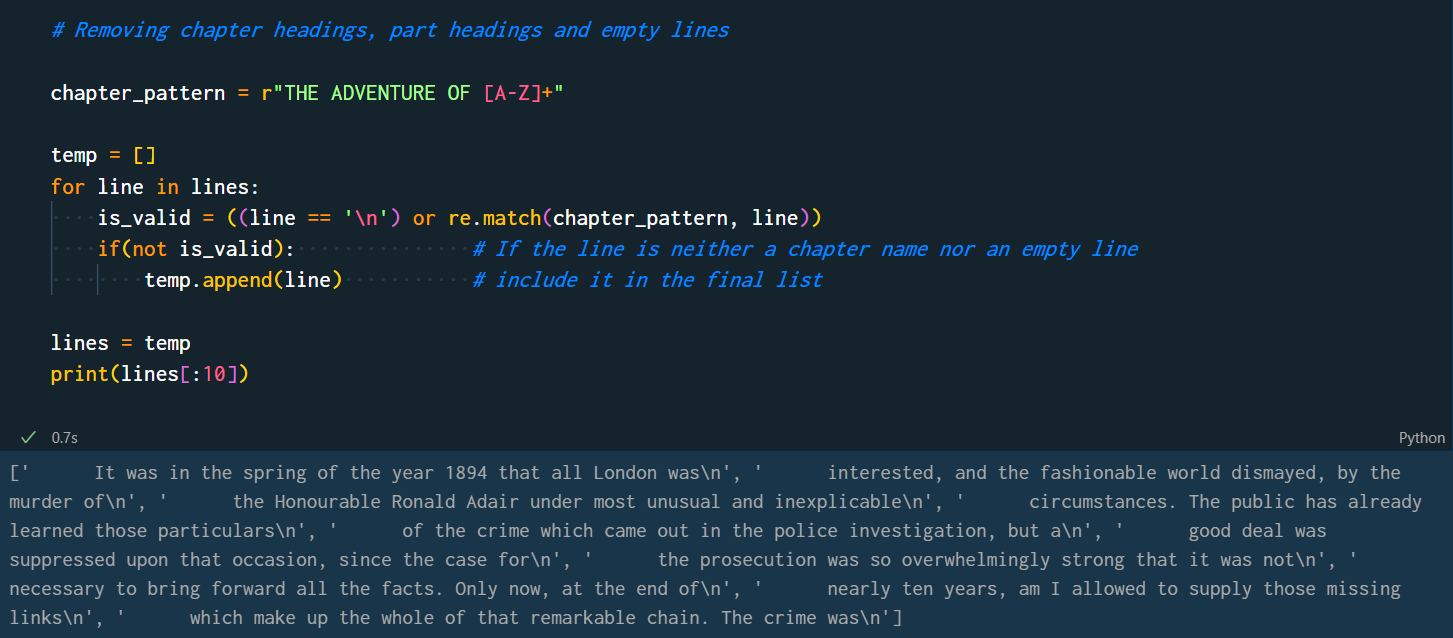


## 

## Removing Running Words

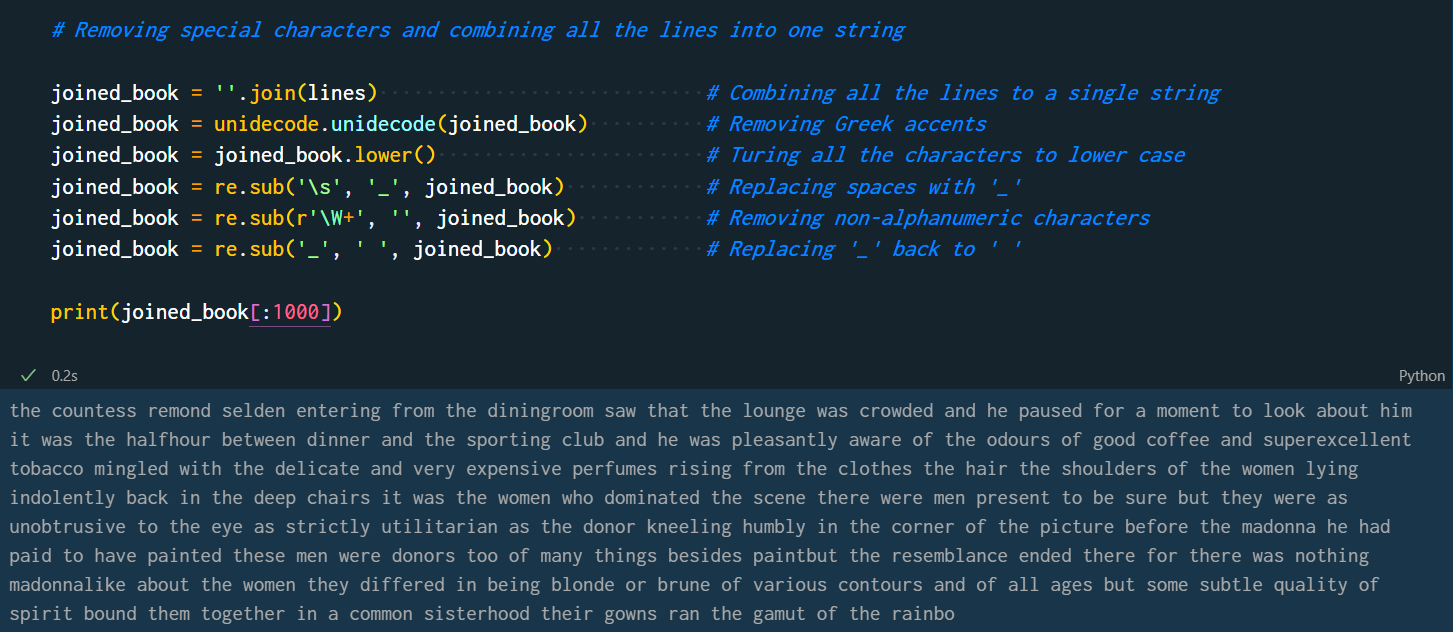
Running words include words that are not part of the main content like chapter headings and part headings. This can be achieved by regex matching.

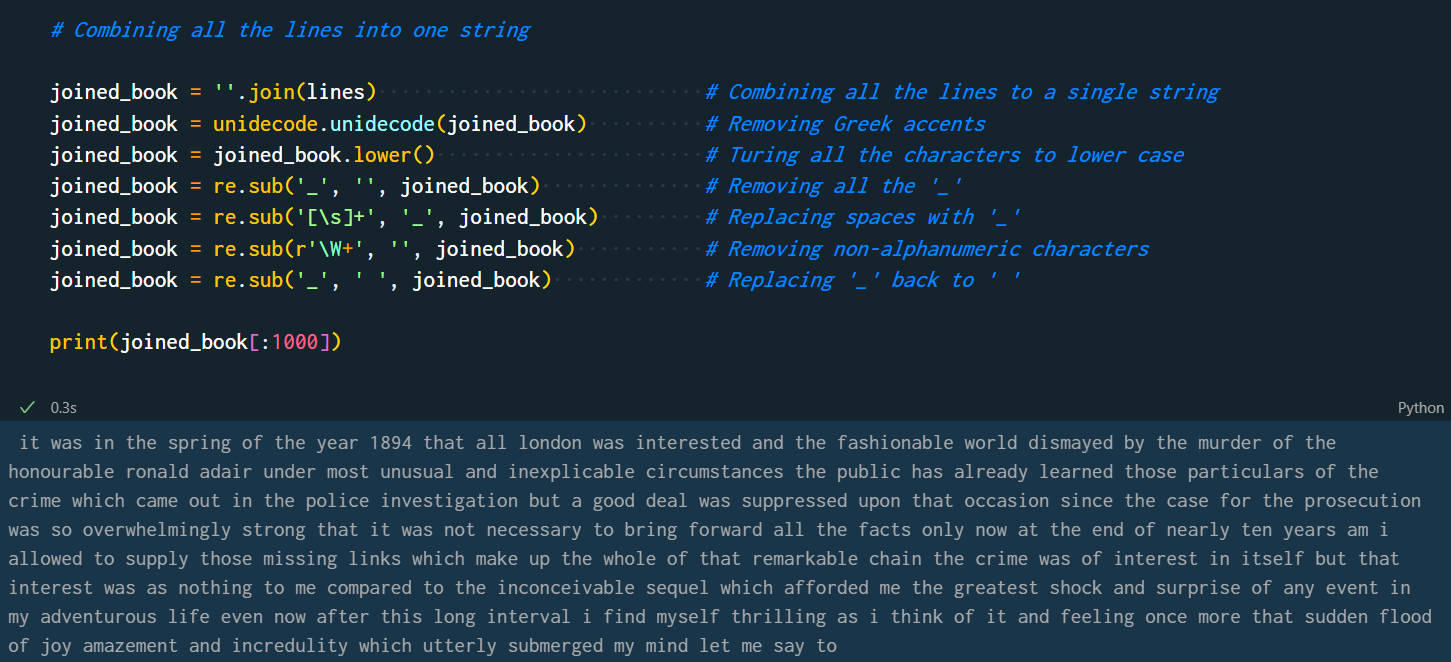




## Concatenating Strings and Removing Special Characters & Spaces

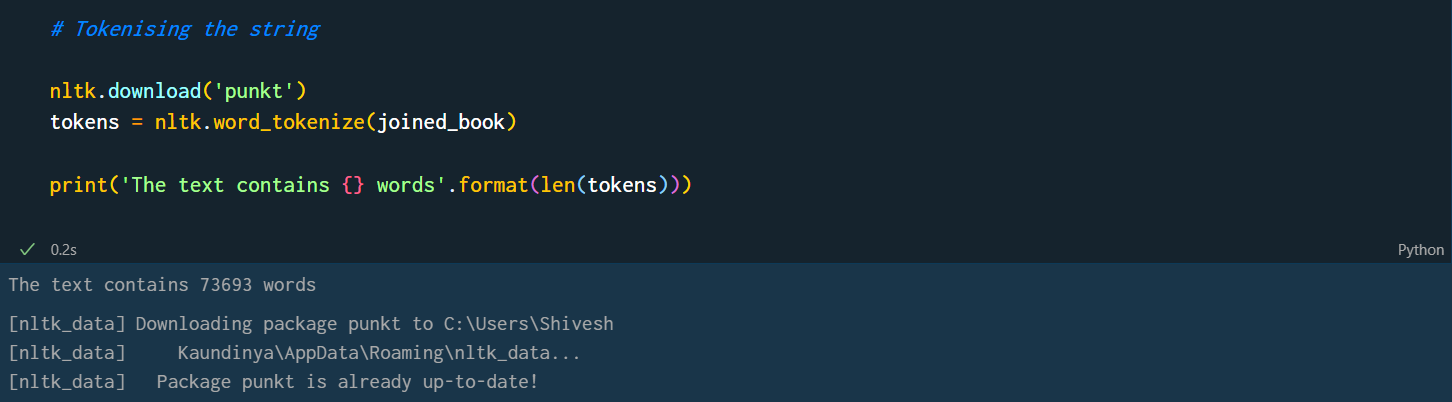
Now we need to remove extra spaces, special characters, Greek accents and merge all the lines to create one string. We will first concatenate all the lines using the string’s join method and then use regex to remove unwanted characters.

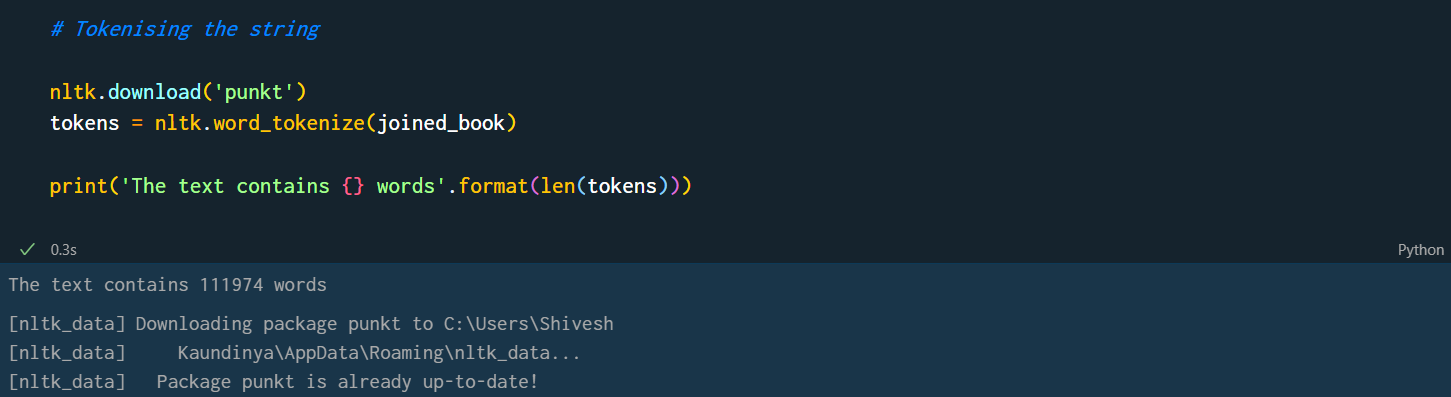




## Tokenizing

The final part of pre-processing is tokenization. It refers to separating all the words into an array. This makes it easier for the computer to perform tagging. It can be done using nltk’s word\_tokenize method.



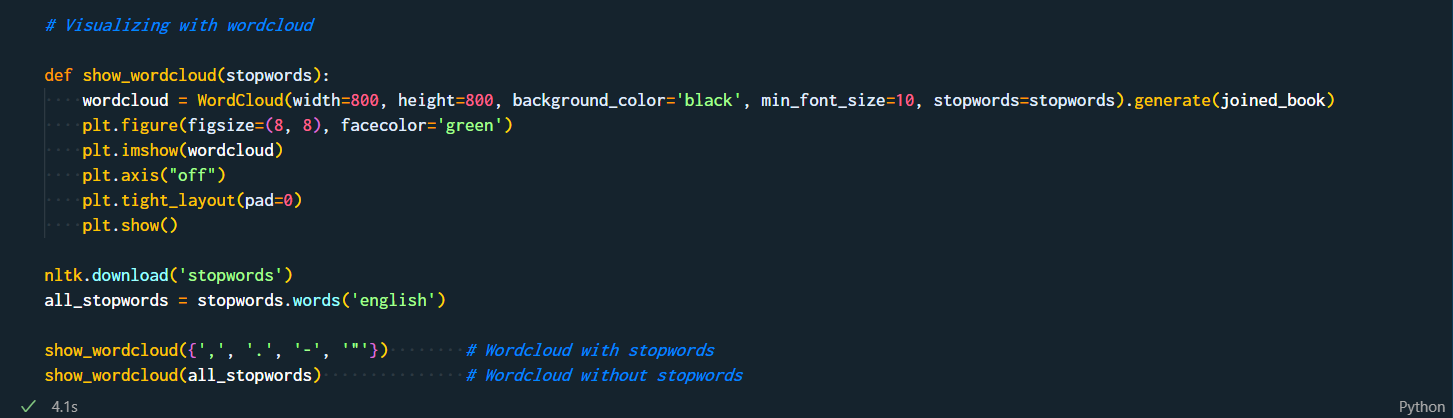


# VISUALIZATION

## Wordcloud

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analyzing data from social network websites.

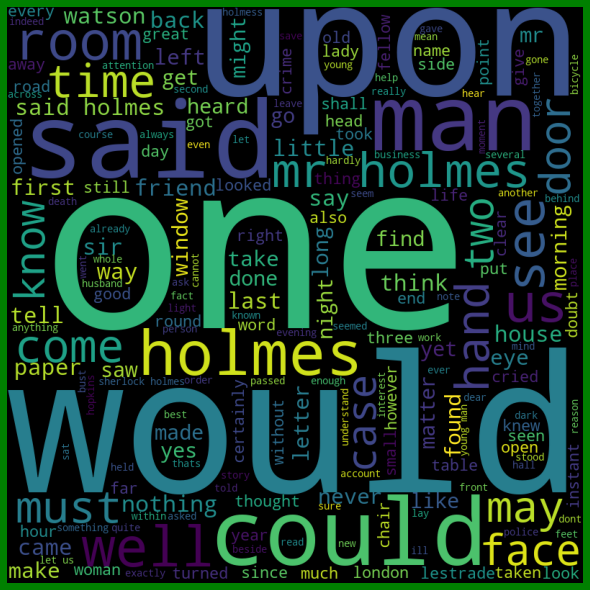
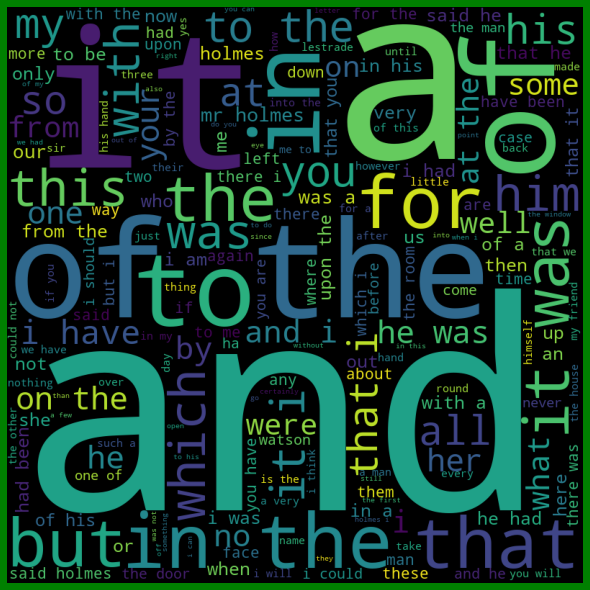
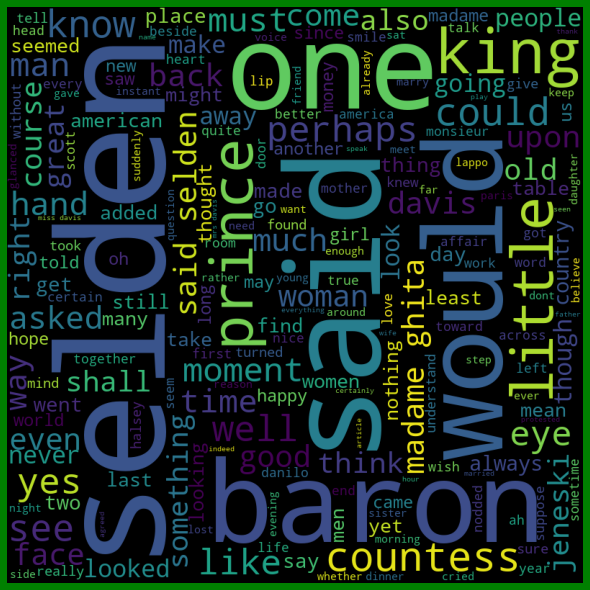
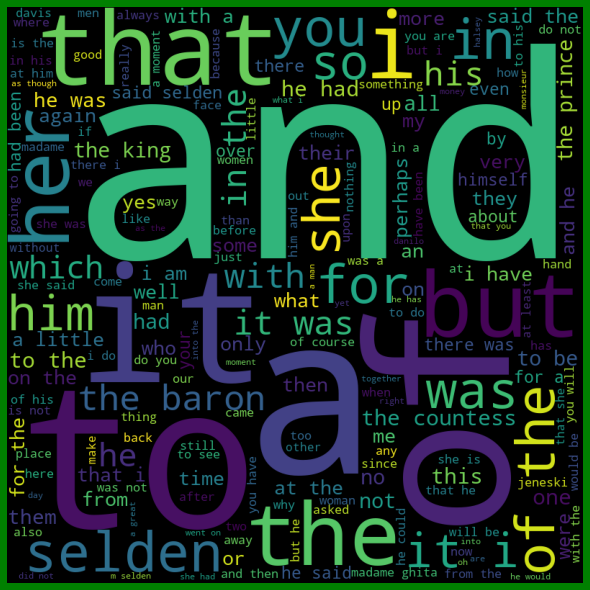
We will be creating the wordcloud both with and without stop words. In python, it can be constructed quite easily using wordcloud and matplotlib libraries.



The wordclouds are as follows:

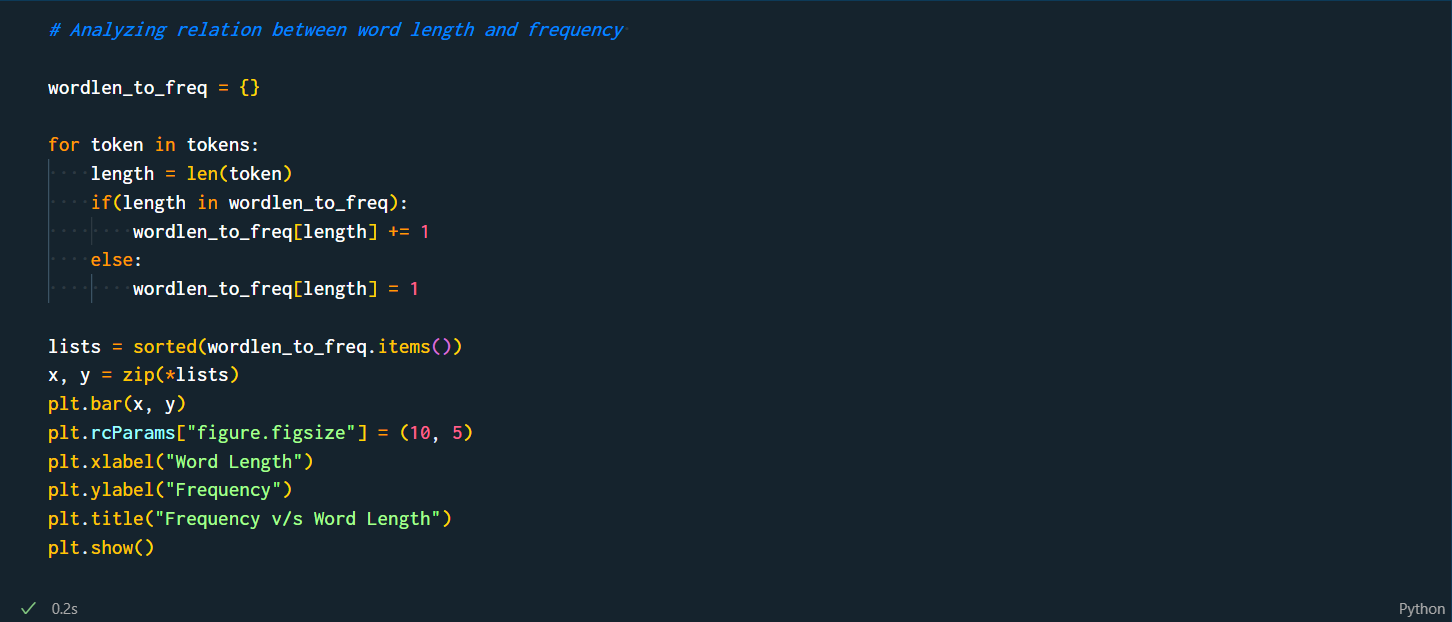
* Top Left: Wordcloud of first text without removing the stop words
* Top Right: Wordcloud of first text after removing the stop words
* Bottom Left: Wordcloud of second text without removing the stop words
* Bottom Right: Wordcloud of second text after removing the stop words

After comparing the wordclouds of texts with and without stop words for both the texts, it appears that most of the most frequent words to appear are stop words. In both the texts, ‘and’, ‘the’, and ‘but’ are quite common occurrences.

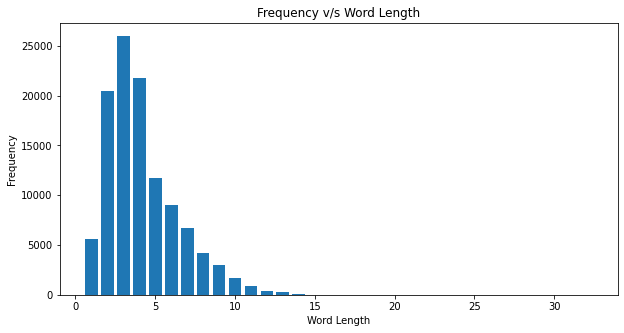


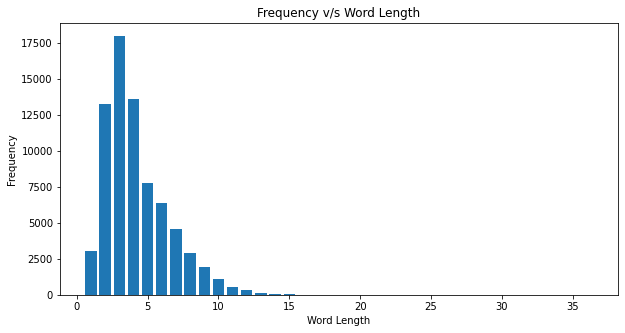
## Frequency Distribution v/s Token Length

Now, let us look at the relationship between the length of a token and its frequency. We can do this by creating a dictionary mapping token length to frequency. Then loop over the list of tokens and increment the frequency corresponding to the length of the token in the dictionary. After that, this frequency distribution can be visualized using matplotlib’s bar function that creates a bar graph.



The bar graphs obtained for both the texts are as follows:





Here, we can observe that in both the texts extremely small, as well as, large words are rare. The frequency first increases and then decreases. Words of length 3 occur most frequently.

Combining this knowledge with the wordclouds above, it can be said that *the most common words used are ‘and’, ‘the’, ‘but’, etc*.

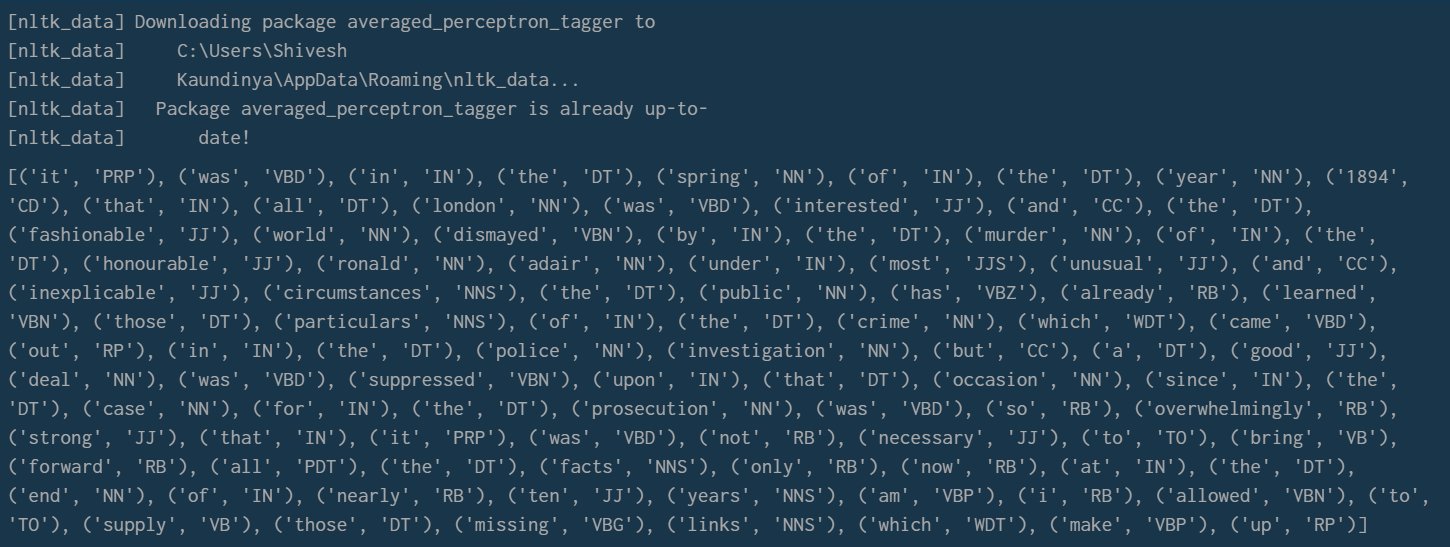
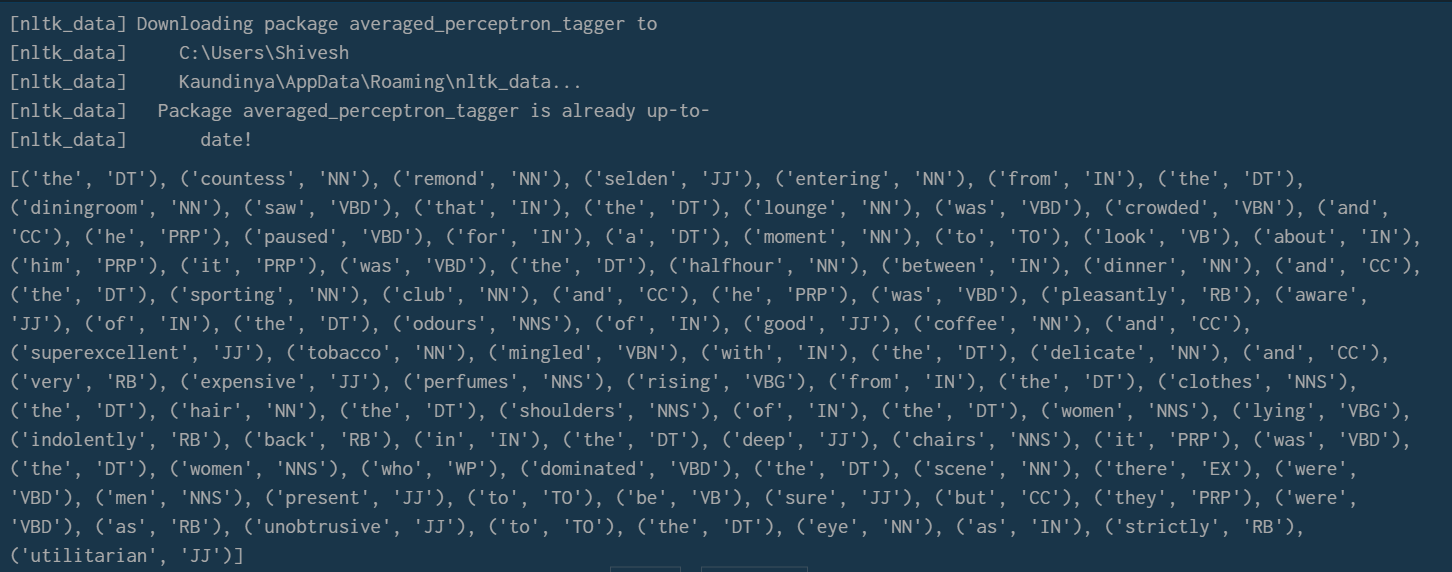
# PART-OF-SPEECH TAGGING

Part-of-speech tagging (or PoS tagging), also called grammatical tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context. It is harder than just having a list of words and their parts of speech, because some words can represent more than one part of speech at different times, and because some parts of speech are complex or unspoken.

We will be using nltk’s pos\_tag method to tag our input text. The tagger used is ‘averaged perceptron tagger’.



The output for both the texts is as follows:



Now that we have our corpuses tagged, we try and visualize this result to get a better understanding and derive inferences.

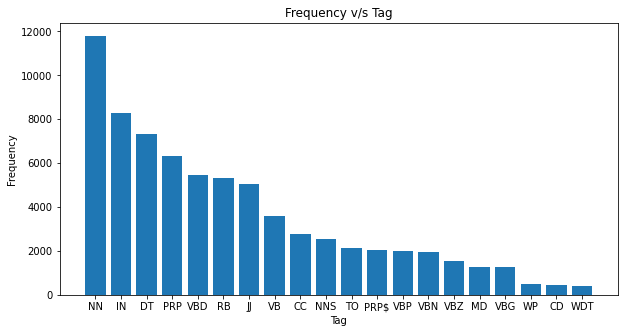
Hence, we can plot a bar graph between the 20 most frequent tags and their frequency.

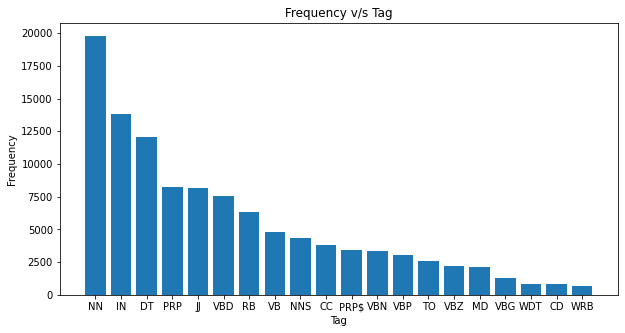
For this purpose, we will first create a dictionary that will have the tags as the keys and their corresponding frequencies as the values. Then, we will iterate through the output list returned by the pos\_tag method and increment the frequency of the tag it contains in the dictionary. Finally, we will sort the dictionary by its values in descending order and pick the top 20 entries from it. Once, we have the dictionary, we can easily plot a bar graph for it using matplotlib.

The code for it is as follows:



The resulting bar graphs are:





These plots show that *nouns are the most common words used* in both texts.

# CONCLUSION

Through this project, we were able to learn various new concepts such as wordclouds, tokenization, PoS tagging, etc. We also learned about new techniques to implement these concepts and gain insights from the data.

We were also able to derive a few inferences on the texts:

* Smaller words are more frequent as compared to larger words.
* The most common word length is 3. This is evident by frequent occurrences of words like ‘and’, ‘the’, ‘was’, etc.
* Among all the kinds of words used, nouns are the most common among them. It is followed by prepositions and determinants.