NATURAL LANGUAGE PROCESSING

Team Infinity

Aarya Garg (20UCS002)

Harshal Jain (20UCS077)

Suhani Sharma (20UCS202)

Yash Jain (20UCS237)

Course Instructor

Dr. Sakthi Balan Muthiah

Departmentof Computer Science Engineering

The LNM Institute of Information Technology

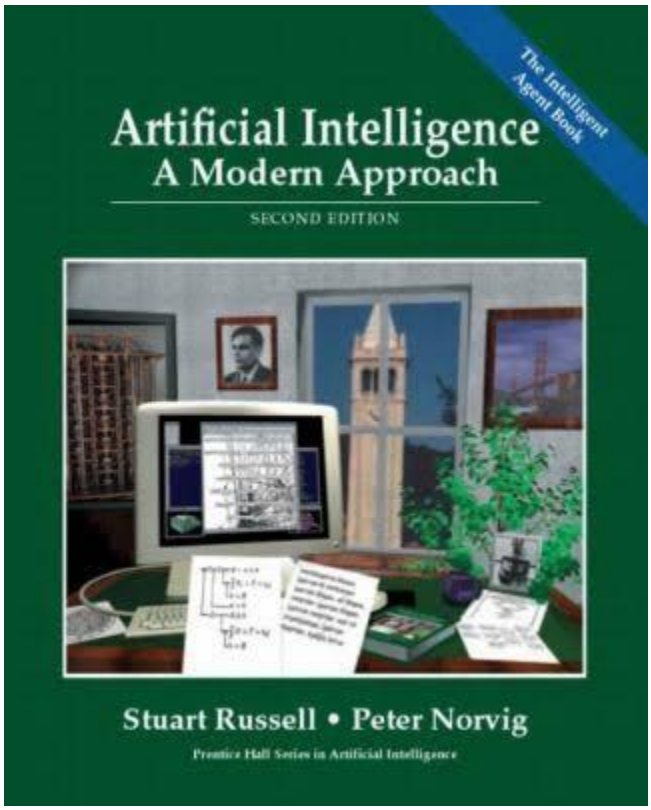
Project Round 1

Github Link:-https://github.com/jYash2309/NLP_PROJECT_ROUND-1

Overview

In this project, we will be analyzing the textbook by Russell and Norvig. We will next proceed and apply POS Tagging to the textbook. With the aid of Python libraries, we will use NLP frameworks to accomplish all of this.

Book used



Artificial Intelligence: A Modern Approach

Textbook by Peter Norvig and Stuart J. Russell

Goals

1. Import the text from the book in text format and call it 'text'.

   (Tool used https://cloudconvert.com/pdf-to-txt)
2. Pre-process the text
3. Tokenize the text
4. Analyze the frequency distribution of tokens in the text
5. Create the Word Cloud using tokens
6. Remove the stopwords from text and again create a word cloud.
7. Compare frequencies with word clouds after removal of stopwords.
8. Analyzing the word length and evaluating the relationship between the word length and frequency for T1.

9. Do PoS Tagging for text and Get the distribution of various tags.

Specifications

Python Libraries used in this project

- Pandas - Used for data manipulation and analysis
- NLTK - Used for Tokenizing, Lemmatization and Removing Stopwords
- Re - Used to remove URLs and Decontract Contractions in English Language
- Wordcloud - Used to create WordClouds from Tokenized Data
- Inflect - Used to replace numbers with words
- Maplotlib - Used to Visualize our text data

Description Of Data

After converting the pdf into txt format the text looks like this

```
In which we try to explain why we consider artificial intelligence to be a subject most worthy of study, and in which we try to decide what exactly it is,
ARTIFICIAL   more complicated than itself. The field of artificial intelligence, or AI, goes further still: it INTELLIGENCE attempts not just to understan
AI is one of the newest fields in science and engineering. Work started in earnest soon after World War II, and the name itself was coined in 1956. Along
AI currently encompasses a huge variety of subfields, ranging from the general (learning and perception) to the specific, such as playing chess, proving m
intellectual task; it is truly a universal field. 1.1 WHAT IS AI?
We have claimed that AI is exciting, but we have not said what it is. In Figure 1.1 we see eight definitions of AI, laid out along two dimensions. The def


                                                                                                    1


21
2                                                                                                                                  Chapter 1.         Introduction


                          Thinking Humanly                                                  Thinking Rationally

                          “The exciting new effort to make comput-        “The study of mental faculties through the
                          ers think ...machines with minds,inthe use of computational models.”
                          full and literal sense.” (Haugeland, 1985)        (Charniak and McDermott, 1985)

                          “[The automation of] activities that we              “The study of the computations that make
                          associate with human thinking, activities        it possible to perceive, reason, and act.”
                          such as decision-making, problem solv-         (Winston, 1992)
                          ing, learning ...” (Bellman, 1978)

                          Acting Humanly                                                      Acting Rationally

                          “The art of creating machines that per-           “Computational Intelligence is the study
                          form functions that require intelligence            of the design of intelligent agents.” (Poole
                          when performed by people.” (Kurzweil,           et al., 1998)
                          1990)
```

Observations

1. There are certain numbers that are not immediately relevant to humans, therefore we must convert them to words.
2. In English, there are a lot of contractions and punctuation, and we need to turn them into useful data.
3. There are several special characters and URLs.
4. Emojis, emoticons, and chat phrases are not available
   #THIS CAN BE IMPROVE

Tasks

1. Importing the book

```
[31] #To open the file
     file = open(r"/content/NlpBook.txt",encoding='utf-8')
     listofwords = file.read().splitlines()
     listofwords = [i for i in listofwords if i!='']
     text = ""
     text = text.join(listofwords)
```

2. Text Preprocessing Steps

For preprocessing we have used following functions: -

a. Filtering the punctuations from the text and converting it into the lower case to make it relevant : -

```
#Defining string which contains the punctuations to reject
punctuations = '''!()-[]{};:'"\,<>./‘’?“”@#$%^&*_~'''
filteredtext = ""
for i in text:
    if i not in punctuations:
        filteredtext = filteredtext + i

#Making the result lowercase
filteredtext = filteredtext.lower()
```

Now the text is filtered and labeled as ‘filteredtext’

3. Tokenizing

Next, we tokenize the filtered text using the function ‘word_tokenize’ from the nltk.tokenize library.

a. Tokenizers divide strings into lists of substrings.
b. This particular tokenizer ‘word_tokenize’ requires the Punkt sentence tokenization model to be installed.
c. This Punkt sentence tokenizer divides text into a list of sentences by using an unsupervised algorithm to build a model for words.

```
tokenisedtext = word_tokenize(filteredtext)
print(tokenisedtext[:10]) #Printing the first 10 tokens of filtered text.
```

Using nltk.FreqDist() function to calculate the frequency of tokens

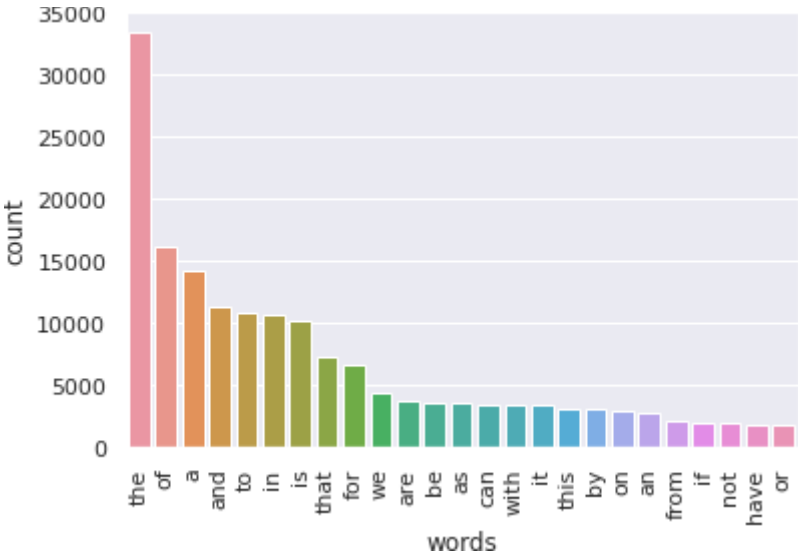
```
frequency_distribution=nltk.FreqDist(filteredtext)

print(frequency_distribution.most_common(10))
freq_dist = list(frequency_distribution)

[(' ', 2327005), ('e', 315885), ('t', 240984), ('a', 216459), ('i', 202808),
```

4. Visualising the frequency distribution of tokens

We use **seaborn** plot to analyze the frequency of the tokens on our dataset. We check the first 25 most occurring tokens.

```
import seaborn as sb
sb.set(style='darkgrid')
dataf=pd.DataFrame(tokenisedtext)
```

```
sb.countplot(x=dataf[0],order=dataf[0].value_counts().iloc[:25].index)
plt.xticks(rotation=90)
plt.xlabel('words')
plt.show()
#Plotting the counts of the most frequent words ordered in descending order of their frequencies.
```

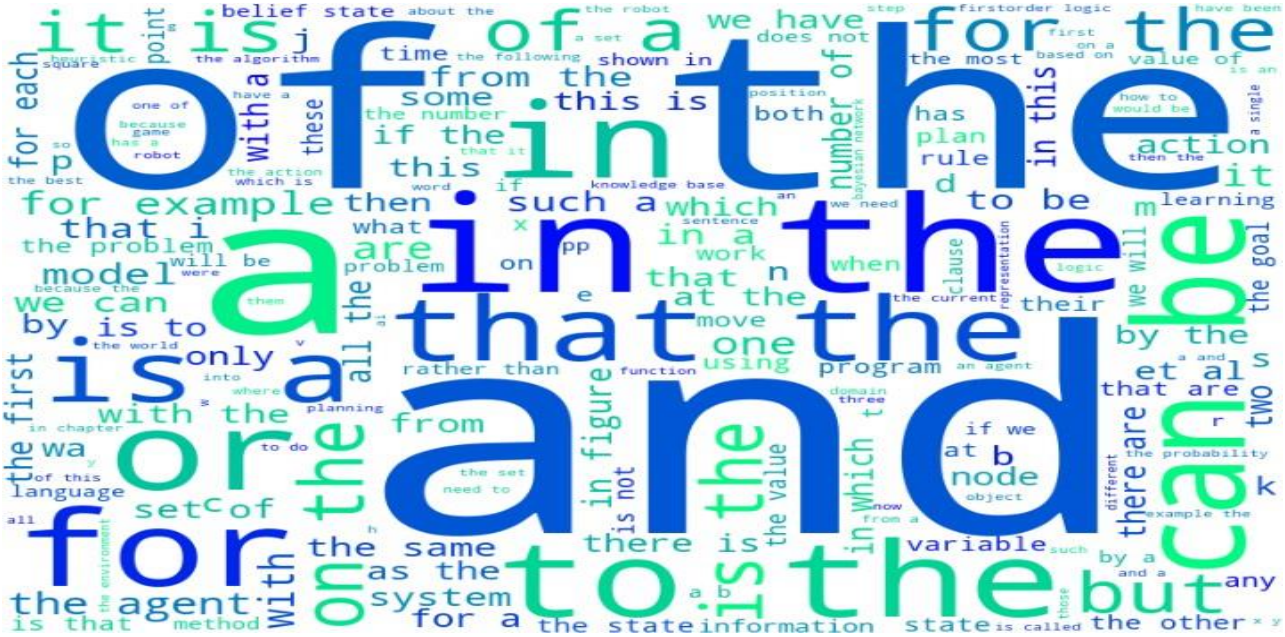


Here most frequent words are plotted with their counts respectively

5. Creating word-cloud with Stop Words

```
Stopwordswordcloud = WordCloud(width = 800, height = 600,
                background_color ='white',
                min_font_size = 10,stopwords = {},colormap='winter').generate(filteredtext)
```

```
plt.figure(figsize = (10,8), facecolor = None)
plt.imshow(Stopwordswordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



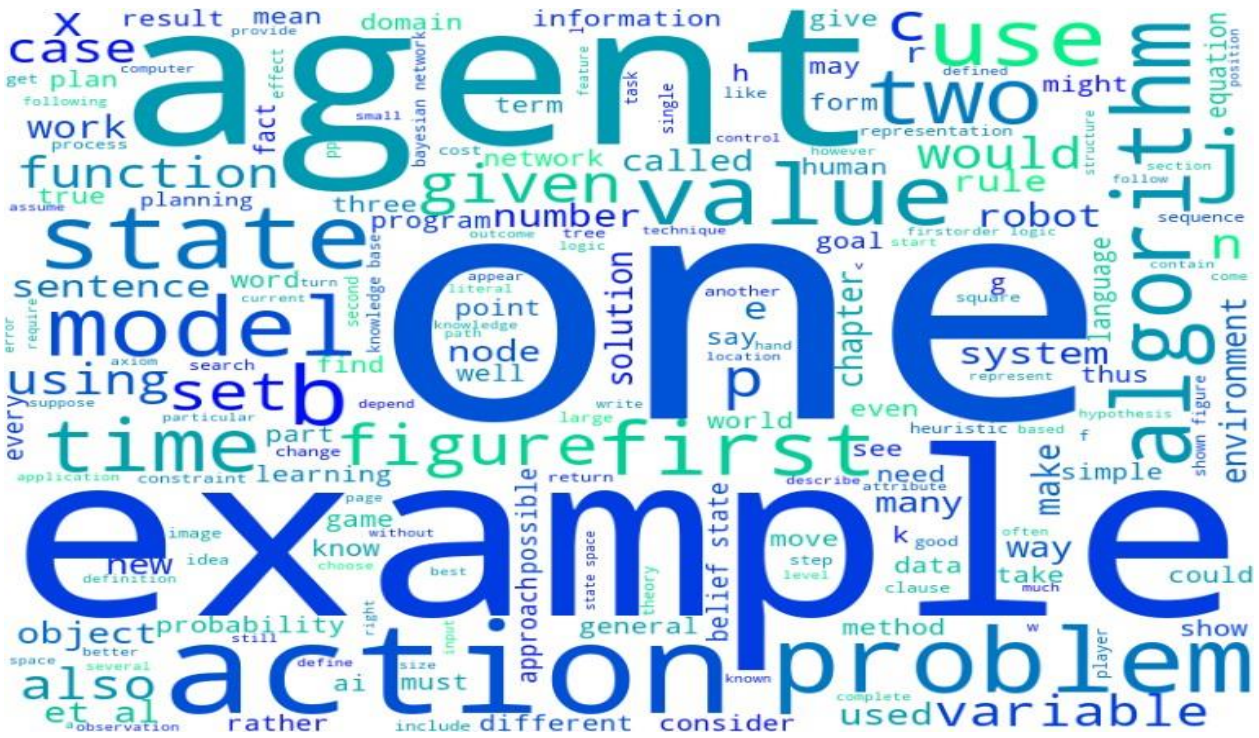
6. Removing the Stop Words and again creating the word cloud

a. Stop word removal :

```
[40] # Stop words need to be removed. So to remove them
     stop_words = set(stopwords.words('english'))
     tokens = word_tokenize(filteredtext)
     final_tokens = [i for i in tokens if not i in stop_words]
     finaltext = "  "
     finaltext = finaltext.join(final_tokens)
```

b. Creating the word cloud

```
# Word cloud after Stopwords removal
wc_withoutStopwords = WordCloud(width = 800, height = 600,
                background_color ='white',
                min_font_size = 10,stopwords = {},colormap='winter').generate(finaltext)

plt.figure(figsize = (12,8), facecolor = None)
plt.imshow(wc_withoutStopwords)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



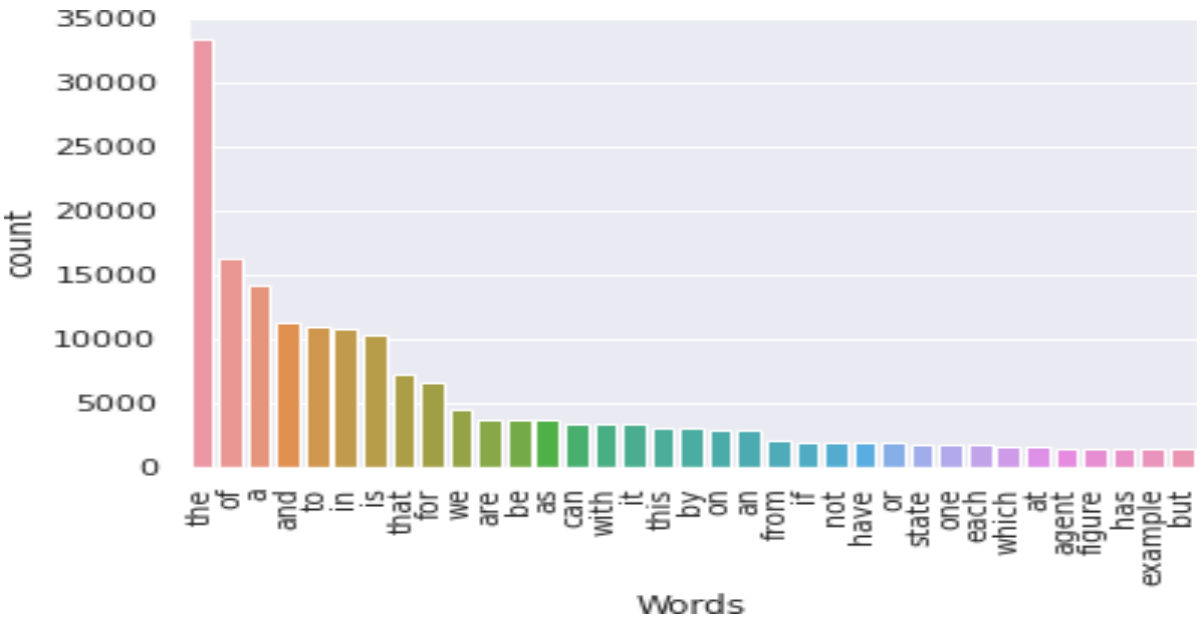
We used the WordCloud function to create word cloud of our dataset without stopwords
After comparing both the word clouds we can observe that previously there were less important words present in the word cloud but after filtering the dataset (i.e. removing the stop words), more important words are given importance in the word cloud.

7. Analysing the frequencies after stopwords removal

```
[42] new_freq_dist=nltk.FreqDist(finaltext)
     print(new_freq_dist.most_common(15))
     freq_dist=list(new_freq_dist)
```

```
     [(' ', 675318), ('e', 246578), ('i', 163375), ('a', 156054), ('t', 153563), ('n', 150480), ('s', 148483), ('r', 136382), (
```

```
new_dataframe=pd.DataFrame(tokenisedtext)
sb.countplot(x=new_dataframe[0], order=new_dataframe[0].value_counts().iloc[:35].index)
plt.xticks(rotation=90)
plt.xlabel('Words')
plt.show()
#Plotting the count after stopwords removal
```



8. Analysing the word length after removing the stop words

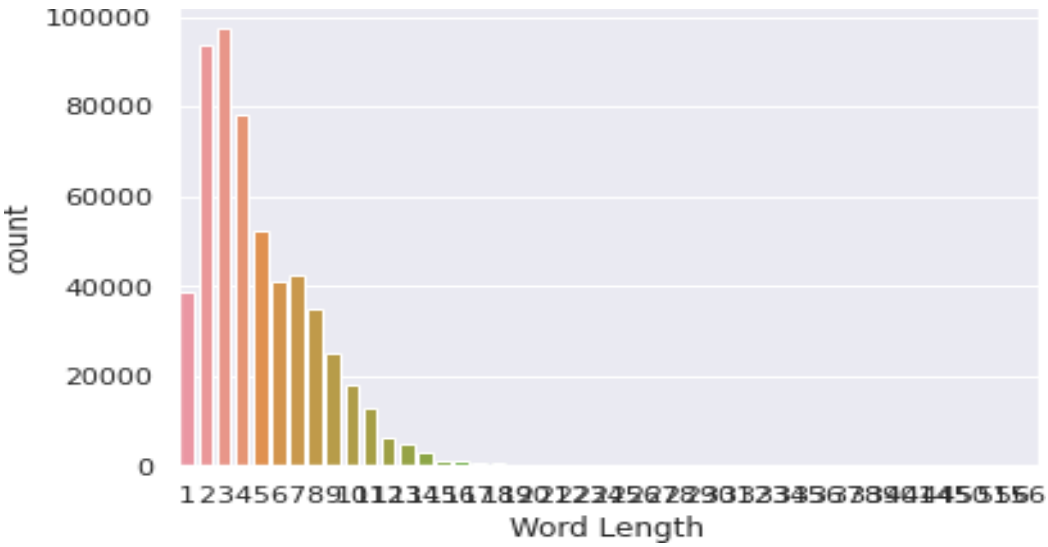
```
[44] length=[]
     for x in tokenisedtext:
      length.append(len(x))
     length[:20]
```

```
[45] frequency_distribution_length=nltk.FreqDist(length)
```

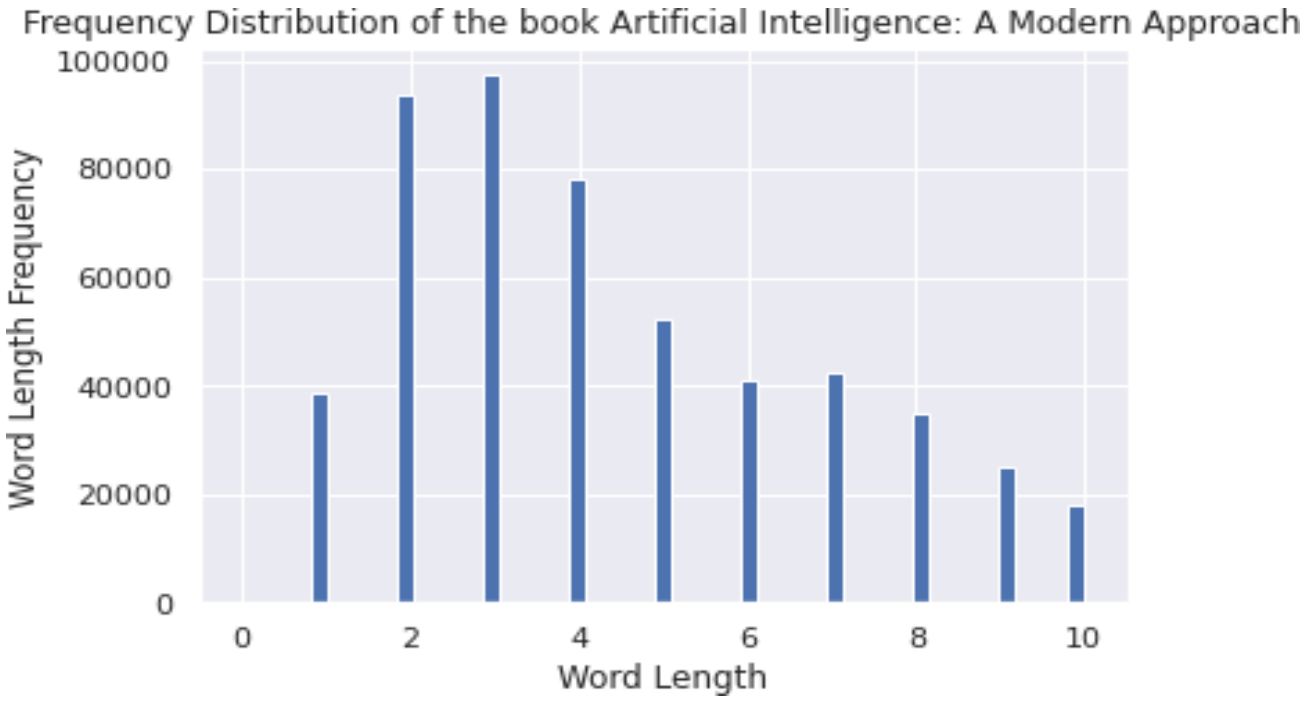
```
[46] frequency_distribution_length
```

```
     FreqDist({3: 97315, 2: 93448, 4: 78358, 5: 52272, 7: 42332, 6: 40948, 1: 38872, 8: 34730, 9: 24855, 10: 18177, ...})
```

```
dataf_length=pd.DataFrame(length)
sb.countplot(x=dataf_length[0])
plt.xlabel('Word Length')
plt.show()
```




```
import numpy as np
bin_size=np.linspace(0,10)

#Finding Wordlength and storing it as a list
wordLength = [len(r) for r in tokenisedtext]

#Plotting histogram of Word length vs Frequency
plt.hist(wordLength, bins=bin_size)
plt.xlabel('Word Length')
plt.ylabel('Word Length Frequency')
plt.title('Frequency Distribution of the book Artificial Intelligence: A Modern Approach')
plt.show()
```



Relationship between the word length and frequency: Here, we tend to analyse a relationship between the word length and how many words with such word length occurs.

- We first associate a bin for the bar graph using “numpy” library
- Then using len() function we calculate the length of each token
- Then we plot a graph for frequency of such word lengths using matplotlib.pyplot

9. Tagging the Parts of Speech using Treebank tagset

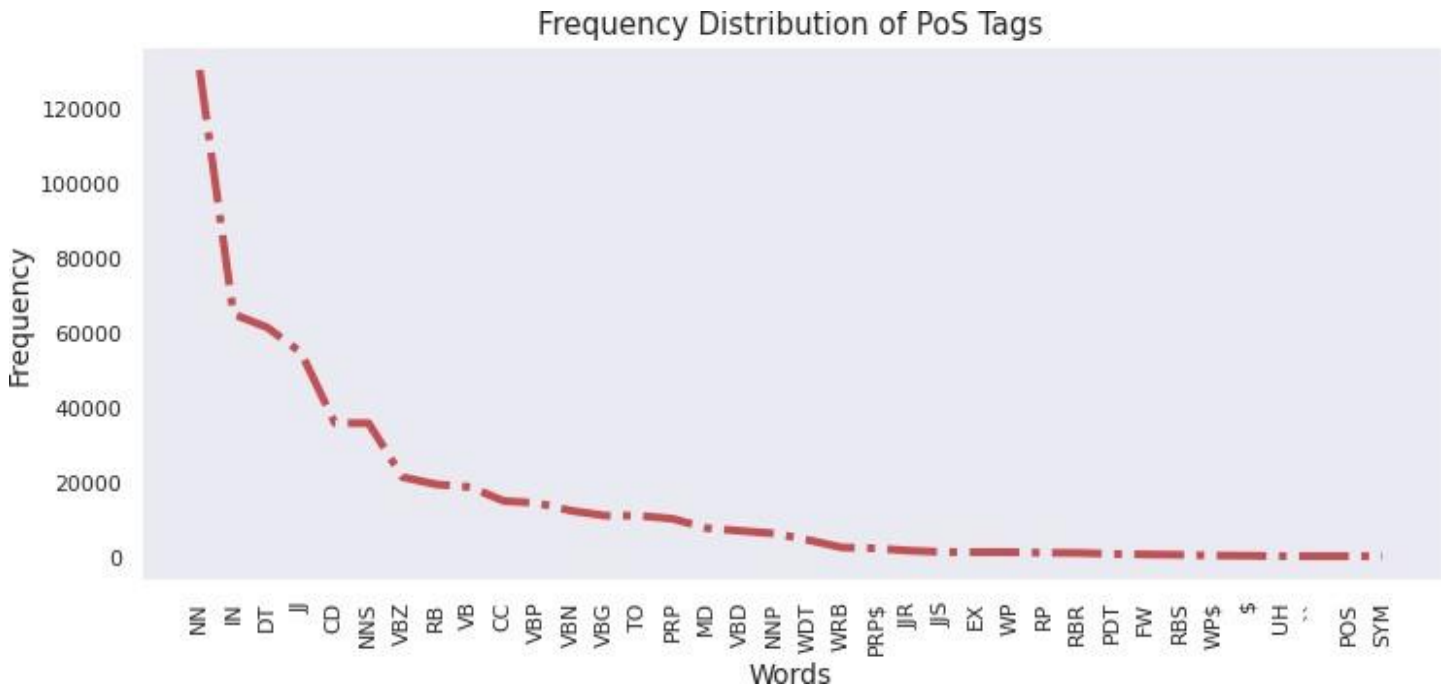
```
tagging = nltk.pos_tag(tokenisedtext)
tagging[:10] #First 10 tags
```

```
[('in', 'IN'),
 ('which', 'WDT'),
 ('we', 'PRP'),
 ('try', 'VBP'),
 ('to', 'TO'),
 ('explain', 'VB'),
 ('why', 'WRB'),
 ('we', 'PRP'),
 ('consider', 'VBP'),
 ('artificial', 'JJ')]
```

```
from collections import Counter
counts = Counter( tag for word,  tag in tagging)
print(counts)
```

```
Counter({'NN': 130391, 'IN': 64912, 'DT': 61347, 'JJ': 54606, 'CD': 35706, 'NNS': 35689, 'VBZ': 21215, 'RB': 19264, 'VB': 18565,
```

```
#Frequency Distribution of PoS Tags

pos_tags_freq = nltk.FreqDist(counts)
pos_tags_freq = {k: v for k, v in sorted(pos_tags_freq.items(), key=lambda item: item[1],reverse=True)}
x = list(pos_tags_freq.keys())[:40]
y = list(pos_tags_freq.values())[:40]
plt.figure(figsize=(12,5))
plt.plot(x,y,c='r',lw=4,ls='-.')
plt.grid()
plt.xticks(rotation=90)
plt.title('Frequency Distribution of PoS Tags',size=15)
plt.xlabel('Words',size=14)
plt.ylabel('Frequency',size=14)
plt.show()
```



After we thoroughly clean our data, we proceed with the text processing part. Here, we have to assign appropriate Part-of-Speech Tags to the words. For this, we use the function “pos_tag( )” from the NLTK library of python.

1. The pos_tag() function uses the Penn Treebank Tag Set, which has 36 tags to assign from to the words.
2. The pos_tag returns a tuple consisting of the token and the tag.

Inference

We applied pos_tagging on the two books using pos_tag function. The pos_tag(words) function uses the Penn treebank as the default tag set as per official documentation.

In The textbook the most frequently occurring POS Tag is ‘NN’ with count 130391 followed by ‘IN’ having count 64912.

Conclusion

In this Round 1 of our project, we performed the tasks of word pre-processing, word tokenization, Word Cloud generation,POS tagging and also deduced many inferences from them about the bookswhile also learning in the process.