NLP Project Round - 1

Github link : https://github.com/Shreya15102/NLP_Project_Round_1

Team Name : Achievers

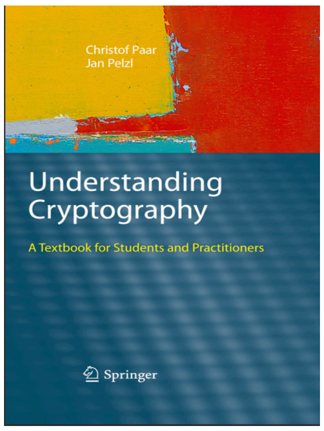
Members:

Shreya Agarwal (20UCS186)
Vansh Patel (20UCS217)
Varad Nandanwankar(20UCS221)

OVERVIEW

In this assignment, we will perform text analysis on our chosen book - **Understanding Cryptography by Christof Paar and Jan Pelzl.** Following that, we'll proceed to apply POS Tagging in the text format. All of this will be accomplished utilizing natural language processing techniques and Python modules.

BOOK USED -


Christof Paar
Jan Pelzl
Understanding Cryptography
A Textbook for Students and Practitioners
Springer

TASK :

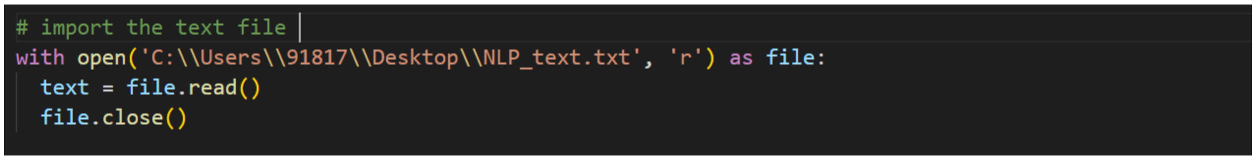
1. Import the text, let’s call it as T1
2. Perform simple text pre-processing steps
3. Tokenize the text T1
4. Analyze the frequency distribution of tokens in T1.
5. Create a Word Cloud of T1
6. Remove the stop words from T1 and then again create a word cloud
7. Evaluate the relationship between the word length and frequency for T1
8. Do PoS Tagging for T1 using anyone of the four tag sets studied in the class and get the distribution of various tags

SPECIFICATIONS:

Python Libraries used in this project

- 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
- Maplotlib - Used to Visualize our text data

Import the Text T1-

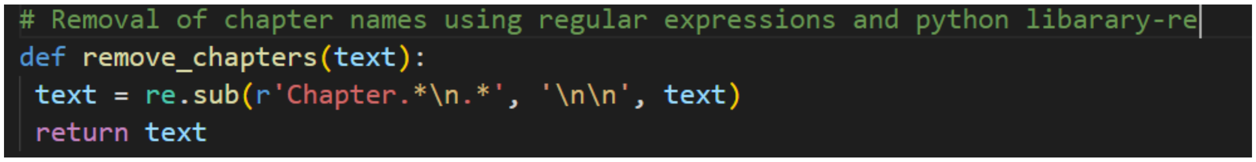


```python
# import the text file
with open('C:\\Users\\91817\\Desktop\\NLP_text.txt', 'r') as file:
    text = file.read()
    file.close()
```

Data Preprocessing Steps-

Removal of Chapter names

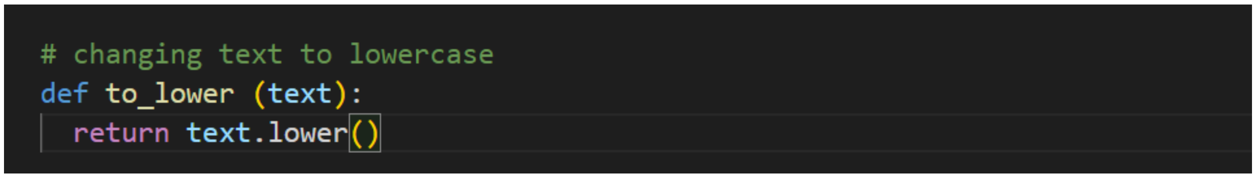
We will do this using a Python Library `re` that will help us apply regular expressions on our data as desired.



```python
# Removal of chapter names using regular expressions and python libarary-re
def remove_chapters(text):
    text = re.sub(r'Chapter.*\n.*', '\n\n', text)
    return text
```

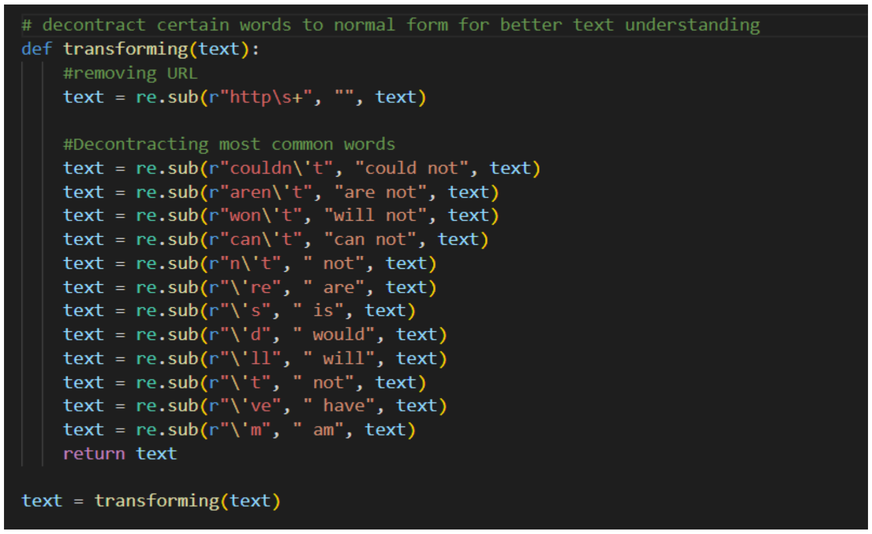
Convert all data to lowercase

We will convert all text data to lowercase, as the case does not contribute much to the meaning of data.



```python
# changing text to lowercase
def to_lower (text):
    return text.lower()
```

Using regular expression to decontract certain words to normal form for better text understanding

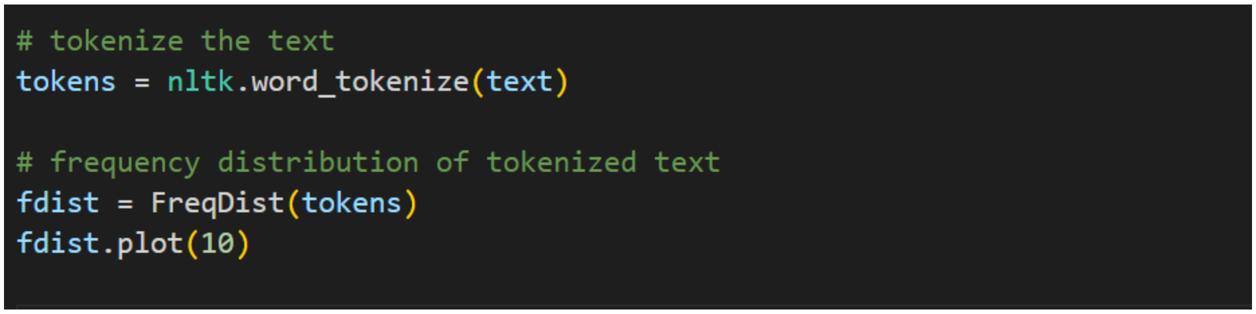


```python
# decontract certain words to normal form for better text understanding
def transforming(text):
    #removing URL
    text = re.sub(r"http\s+", "", text)

    #Decontracting most common words
    text = re.sub(r"couldn\'t", "could not", text)
    text = re.sub(r"aren\'t", "are not", text)
    text = re.sub(r"won\'t", "will not", text)
    text = re.sub(r"can\'t", "can not", text)
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)
    return text

text = transforming(text)
```

Analyze the frequency distribution of tokens in T1

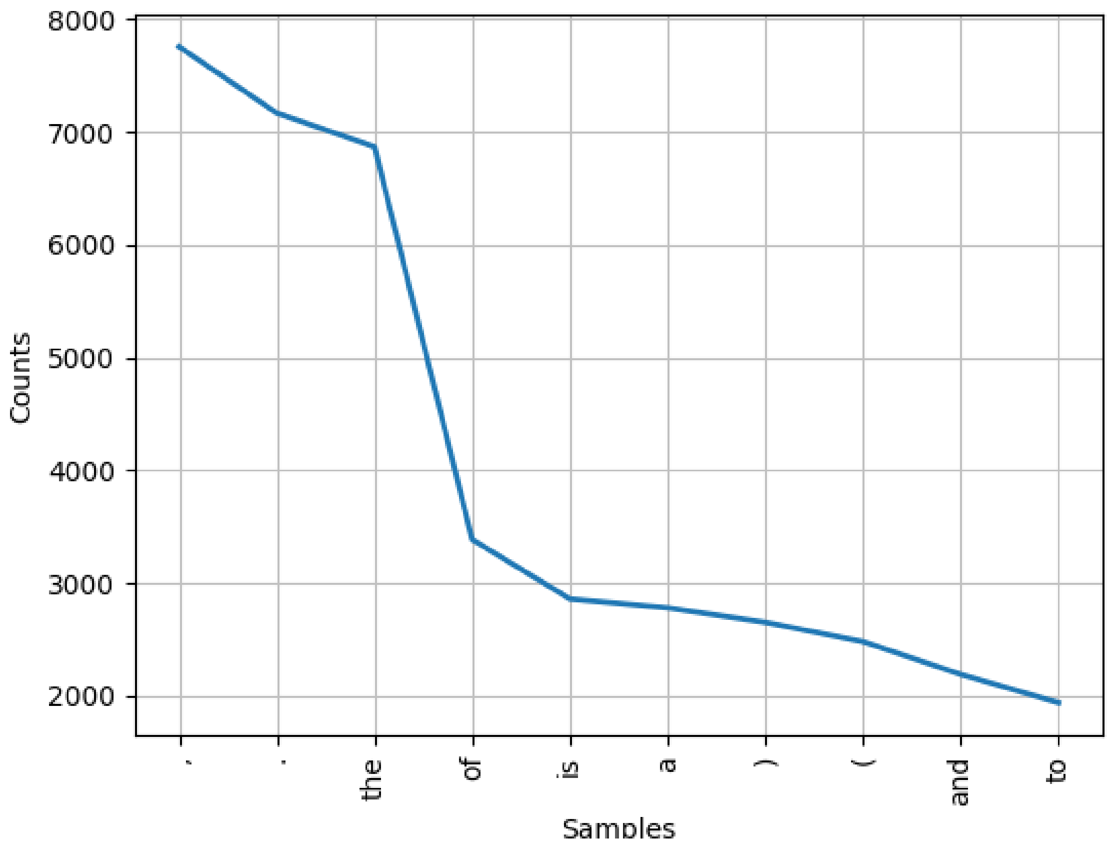
First we tokenize the given data using NLKT module `tokenize` with a function `word_tokenize()` that will help us split a text into tokens. , and then we plot the frequency of top 10 most frequent words.The `FreqDist` **function** gives the user the frequency distribution of all the words in the text.

Tokenize the text



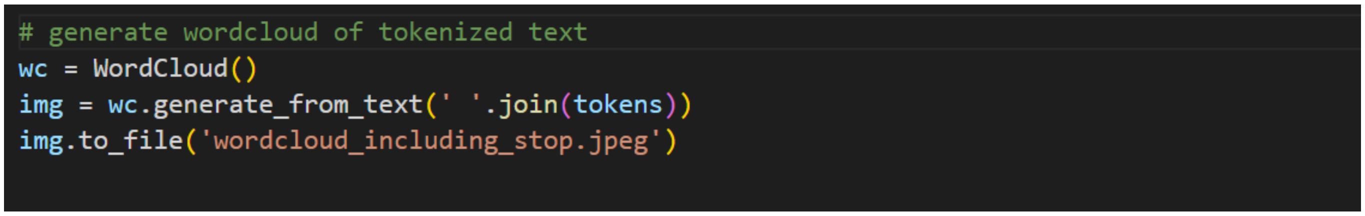
```
# tokenize the text
tokens = nltk.word_tokenize(text)

# frequency distribution of tokenized text
fdist = FreqDist(tokens)
fdist.plot(10)
```

Frequency Distribution-



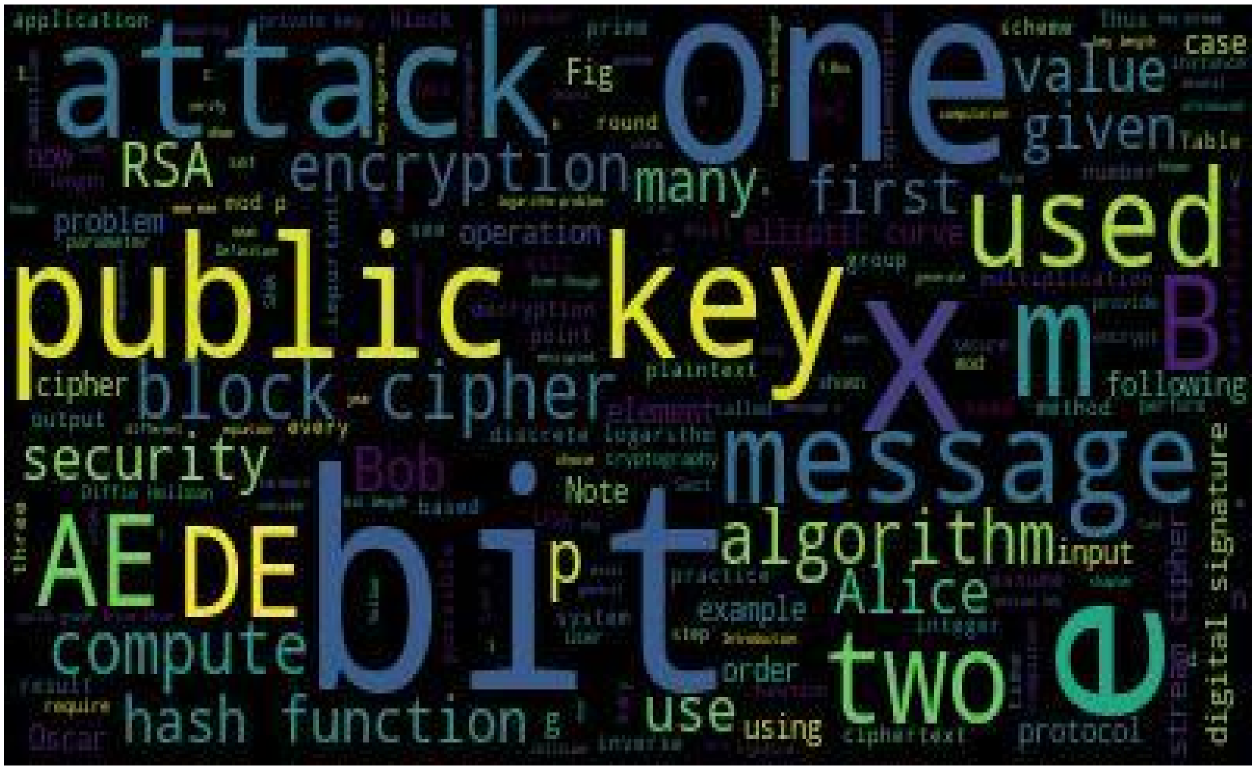
Create a Word Cloud of the tokenized text

For this we take the help of a python library wordcloud and its function WordCloud
It helps in generating wordclouds from a list of Tags from Text data

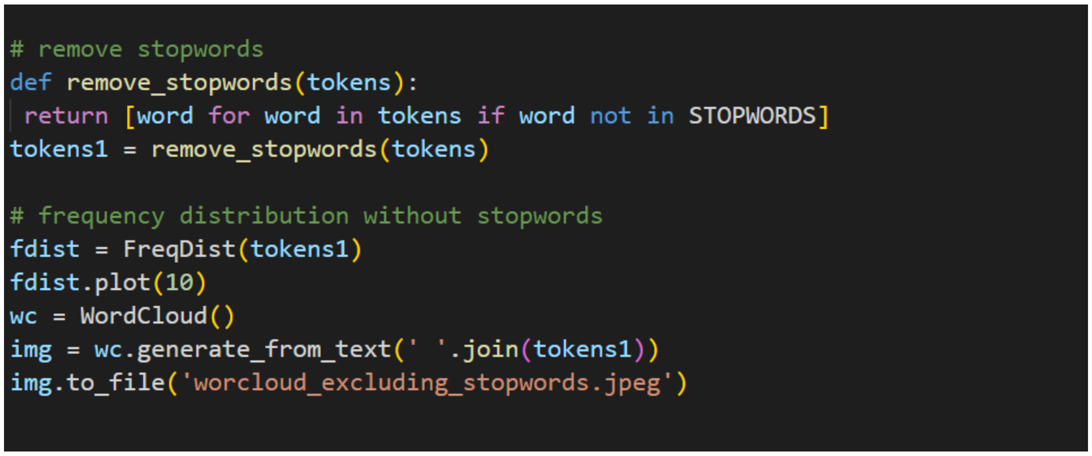


```python
# generate wordcloud of tokenized text
wc = WordCloud()
img = wc.generate_from_text(' '.join(tokens))
img.to_file('wordcloud_including_stop.jpeg')
```

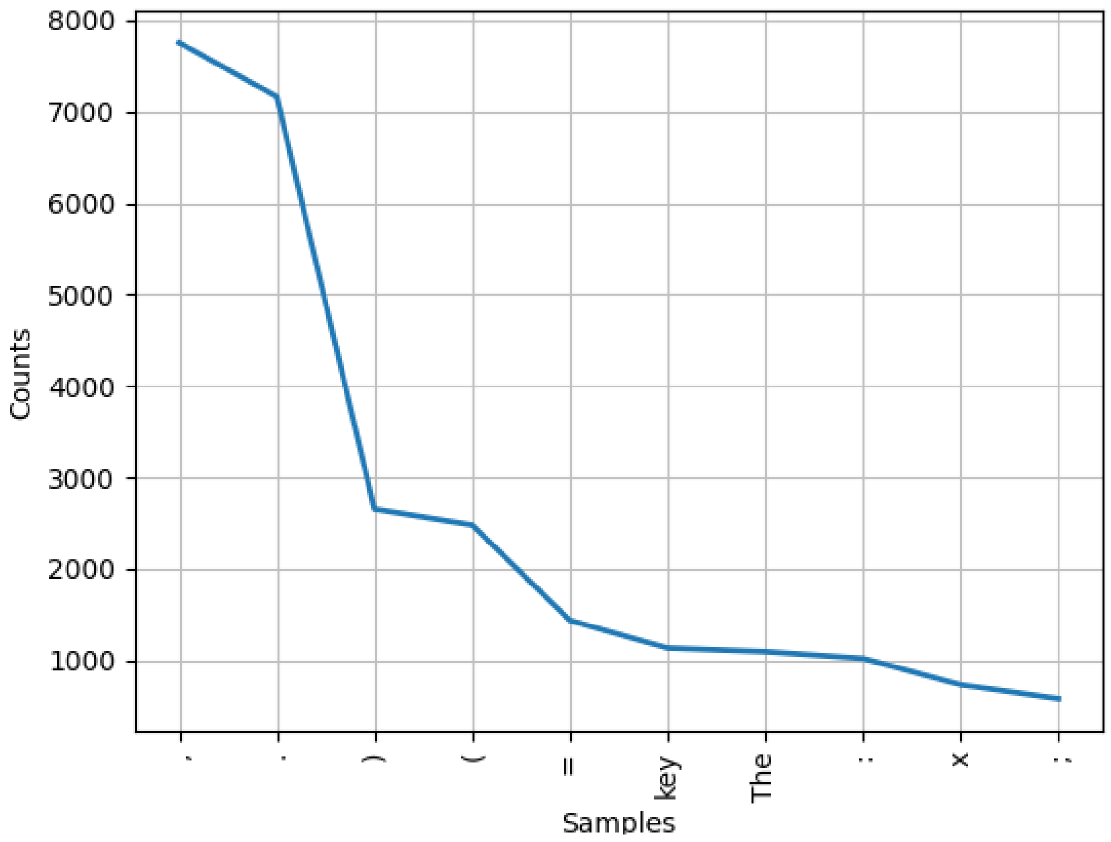
Word Cloud before removing stopwords-



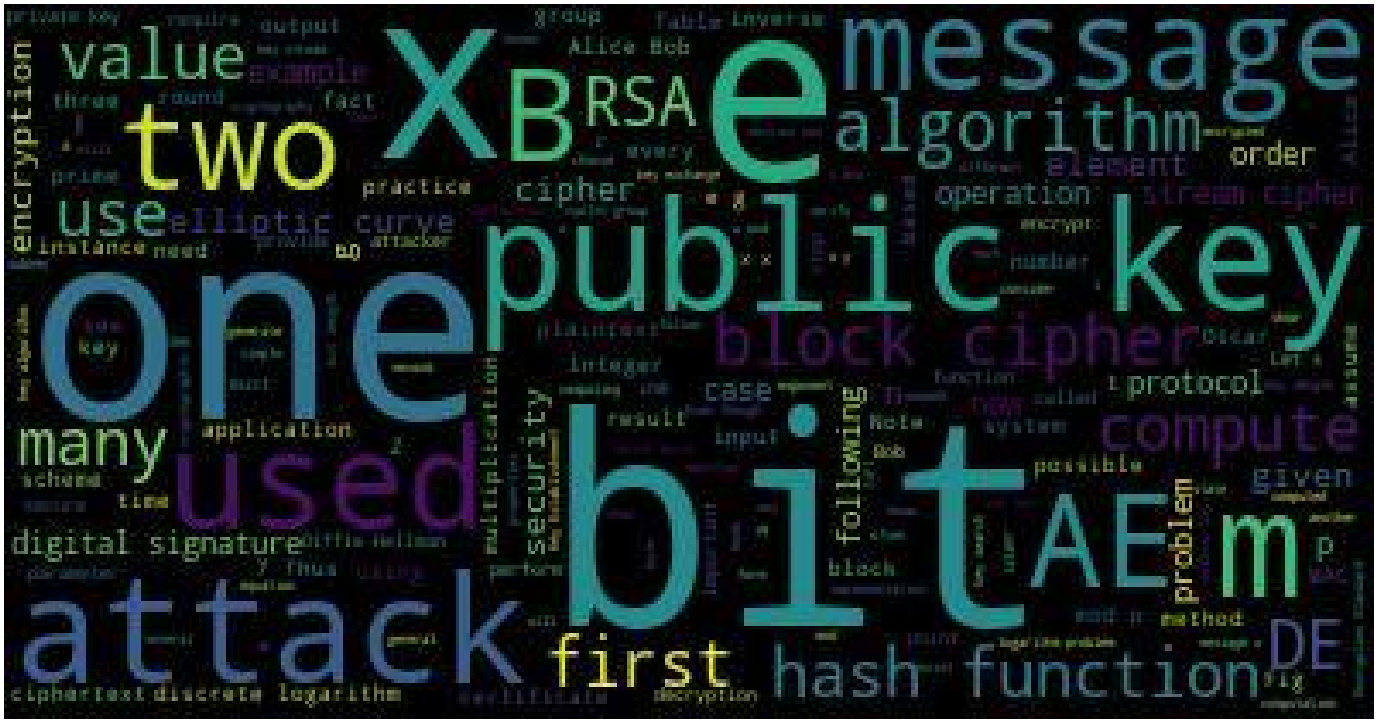
Generating new wordcloud after removing stopwords



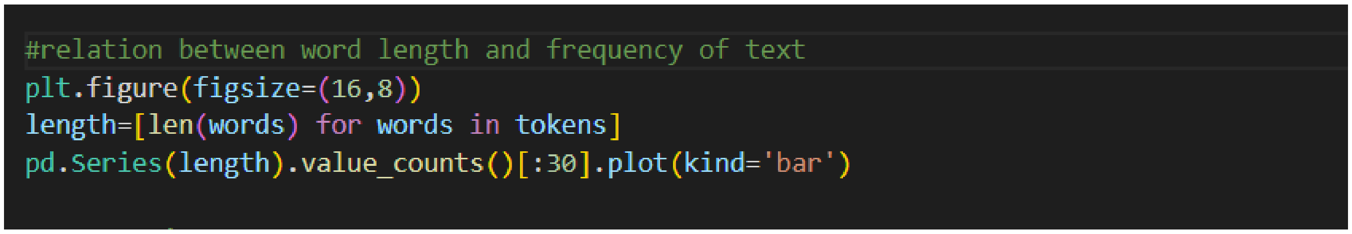
```
# remove stopwords
def remove_stopwords(tokens):
 return [word for word in tokens if word not in STOPWORDS]
tokens1 = remove_stopwords(tokens)

# frequency distribution without stopwords
fdist = FreqDist(tokens1)
fdist.plot(10)
wc = WordCloud()
img = wc.generate_from_text(' '.join(tokens1))
img.to_file('worcloud_excluding_stopwords.jpeg')
```



Wordcloud after removal of stopwords-

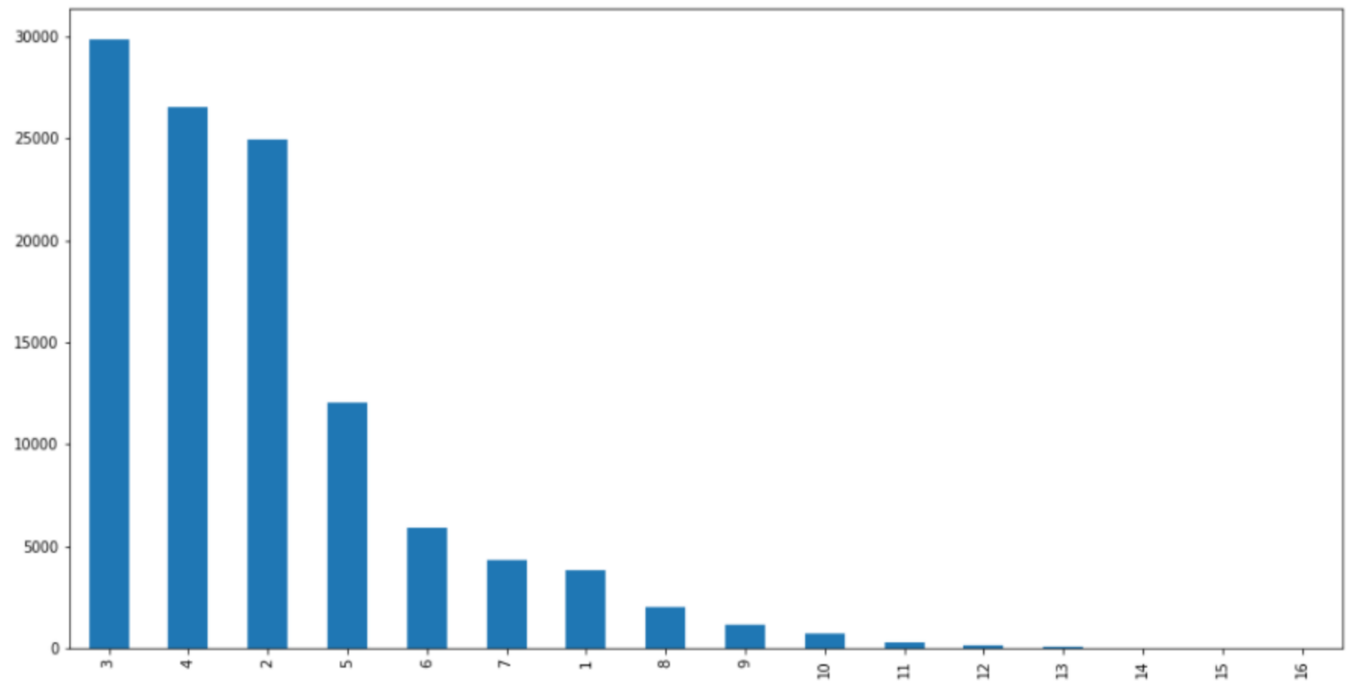


Evaluating the relationship between the word length and frequency of text

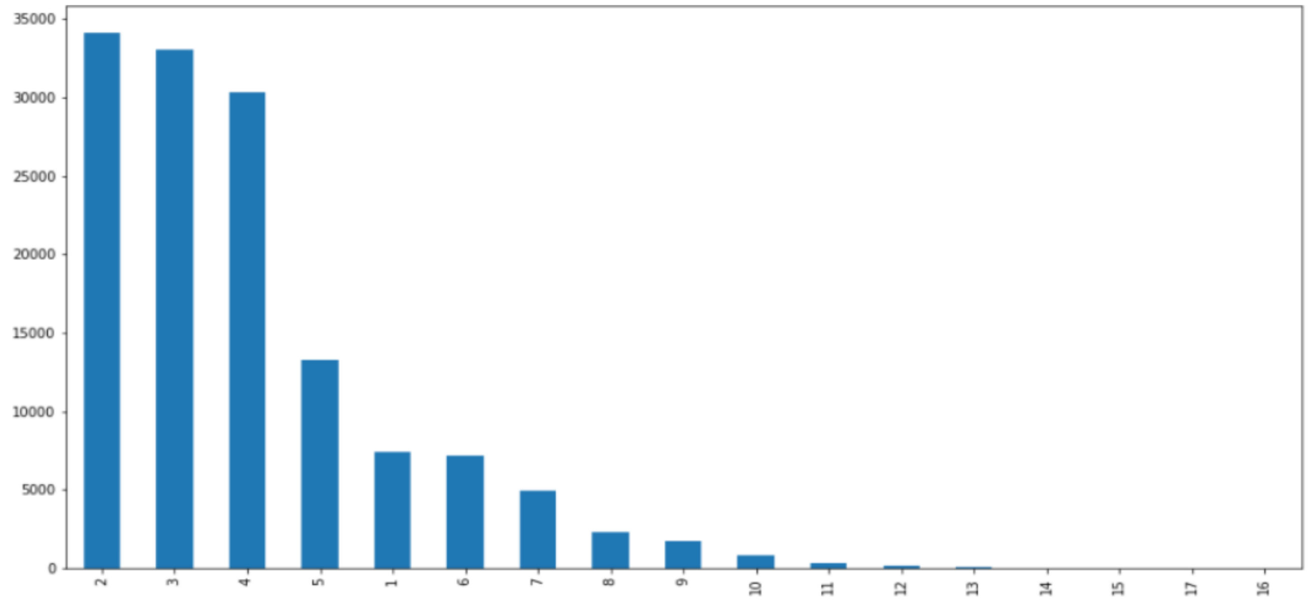


```
#relation between word length and frequency of text
plt.figure(figsize=(16,8))
length=[len(words) for words in tokens]
pd.Series(length).value_counts()[:30].plot(kind='bar')
```

Before removing stopwords



After removing stopwords-

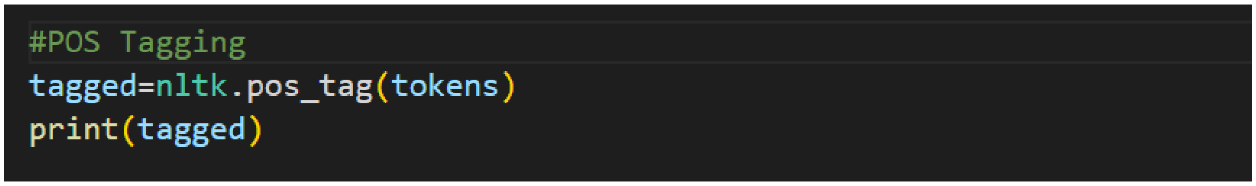


Inference

The number of words of 2 and 3 has been decreased after removing stopwords. This is since stopping words like ‘be’, ‘of’’ have been removed. Apart from that, new comments have emerged, the stop words of length 3,4,5.

Performing POS Tagging

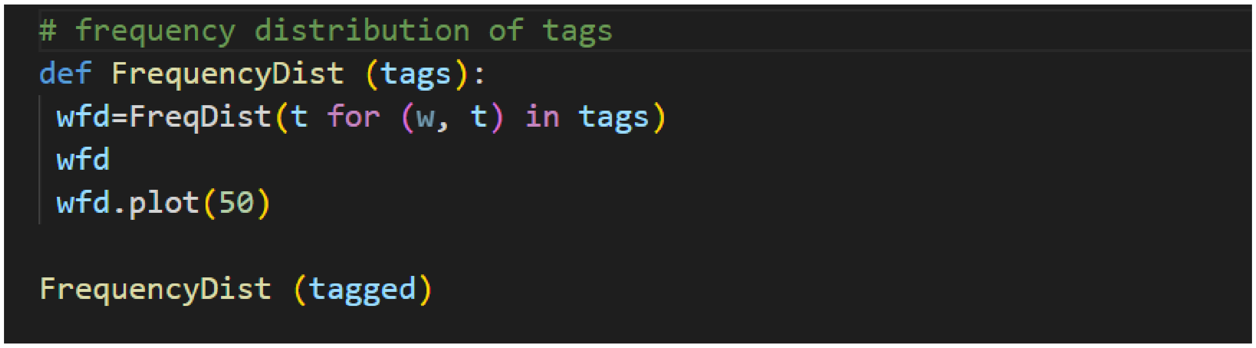
We will now perform the POS Tagging on T1 using inbuilt functions of nltknamely post_tag() which uses Penn Treebank tag set to performPOS tagging.



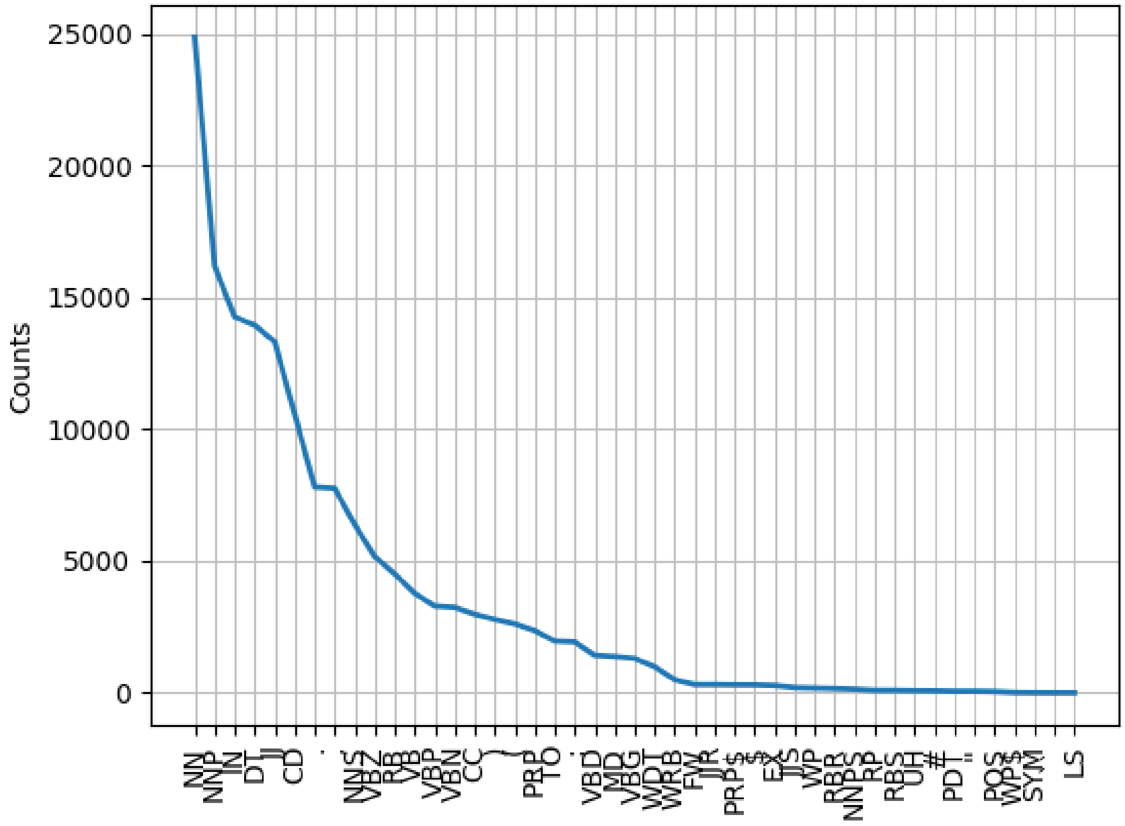
```
#POS Tagging
tagged=nltk.pos_tag(tokens)
print(tagged)
```

```
('Understanding', 'VBG')
('Cryptography', 'NNP')
('Christof', 'NNP'),
('Paar', 'NNP'),
('Jan', 'NNP'),
('Pelzl', 'NNP'),
('A', 'NNP'),
('Security', 'NNP'),
('Textbook', 'NNP'),
('for', 'IN'),
('Students', 'NNP'),
('Practitioners', 'NNP'),
('Foreword', 'NNP'),
('by', 'IN')
('Chair', 'NNP')
('Embedded', 'NNP'),
('Department', 'NNP'),
('of', 'IN')
('Electrical', 'NNP'),
('Engineering', 'NNP'),
('and', 'CC'),
('Information', 'NNP'),
('Sciences', 'NNP')
```

Frequency Distribution of Tags in text T1



```
# frequency distribution of tags
def FrequencyDist (tags):
    wfd=FreqDist(t for (w, t) in tags)
    wfd
    wfd.plot(50)

FrequencyDist (tagged)
```



Inferences

From the above results we infer that the highest occurring tag is ‘NN’ and ‘LS’ tags on the lower frequency side. This is primarily due to removal of stopwords before POS Tagging

Conclusion

We learned how to do Text Preprocessing ,Tokenization and POS Tagging on Text Data, as well as how to use Plots and Visualizations to answer all of the problems presented to us in Round 1 of the NLP Project, and how to make meaningful inferences from it.