Link:- https://github.com/shail10/NLP-project

Language Learners

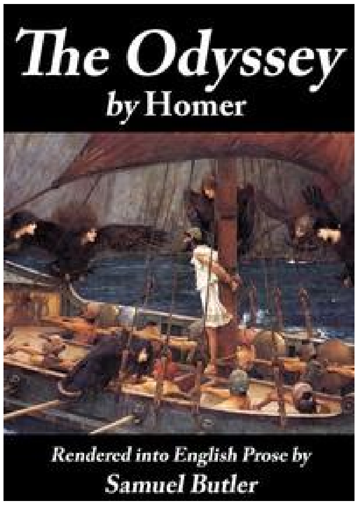
Pratik Gupta (19ucs047)
Raghav R Sharma (19ucs204)
Shail Kardani (19ucs217)

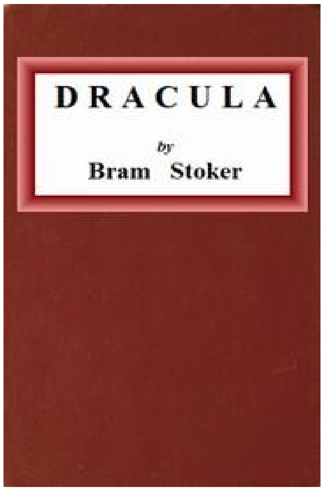
NLP PROJECT - ROUND 1

OVERVIEW

In this project, we perform text analysis on two of our chosen books from Gutenberg. “Dracula” and “Odyssey.” After that, we will apply POS Tagging on both books.

IMPORTED BOOKS

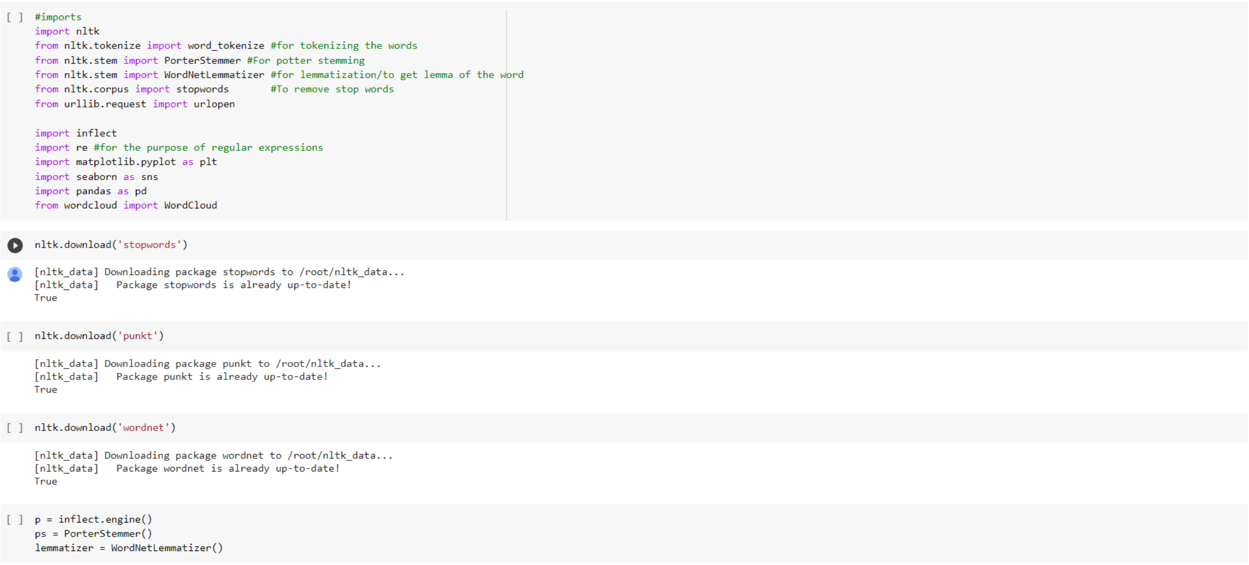




OBJECTIVES

- Import the text, let’s call it T1 and T2
- Perform simple text preprocessing steps and tokenising the text T1 and T2.
- Analyze the frequency distribution of tokens in T1 and T2 separately.
- Create a Word Cloud of T1 and T2 using the token that we have got.
- Remove the stop words from T1 and T2 and then again create a word cloud. Comparison with word clout before the removal of stop words.
- Evaluate the relationship between the word length and frequency for both T1 and T2.
- Do PoS Tagging for both T1 and T2 using any tokenising of the four tag sets studied in the class and get the distribution of various tags.

Libraries Used:-



```
[ ] #imports
    import nltk
    from nltk.tokenize import word_tokenize #for tokenizing the words
    from nltk.stem import PorterStemmer #For potter stemming
    from nltk.stem import WordNetLemmatizer #for lemmatization/to get lemma of the word
    from nltk.corpus import stopwords       #To remove stop words
    from urllib.request import urlopen

    import inflect
    import re #for the purpose of regular expressions
    import matplotlib.pyplot as plt
    import seaborn as sns
    import pandas as pd
    from wordcloud import WordCloud
```

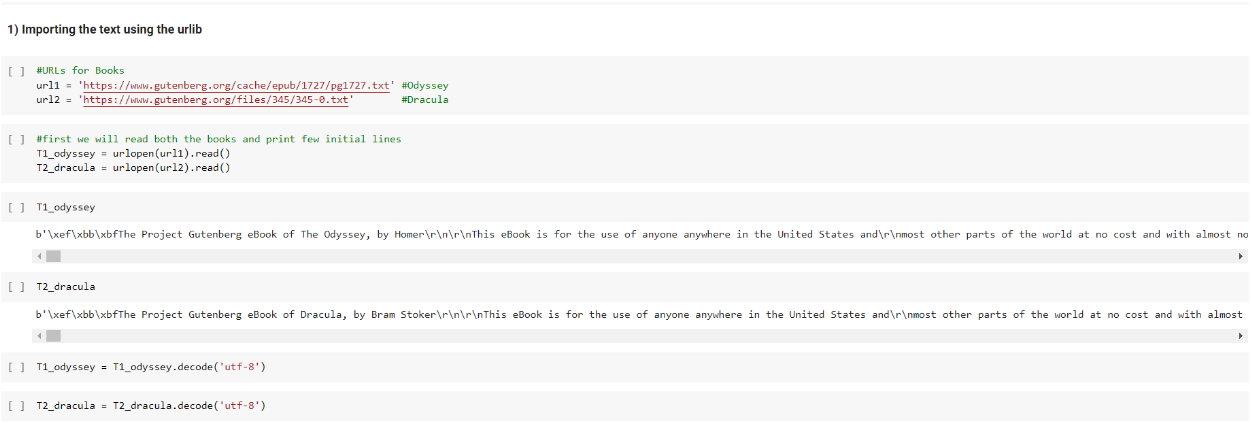
```
    nltk.download('stopwords')

    [nltk_data] Downloading package stopwords to /root/nltk_data...
    [nltk_data]   Package stopwords is already up-to-date!
    True
```

```
[ ] nltk.download('punkt')

    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Package punkt is already up-to-date!
    True
```

```
[ ] nltk.download('wordnet')

    [nltk_data] Downloading package wordnet to /root/nltk_data...
    [nltk_data]   Package wordnet is already up-to-date!
    True
```

```
[ ] p = inflect.engine()
    ps = PorterStemmer()
    lemmatizer = WordNetLemmatizer()
```

Data preprocessing



1) Importing the text using the urlib

```
[ ] #URLs for Books
    url1 = 'https://www.gutenberg.org/cache/epub/1727/pg1727.txt' #Odyssey
    url2 = 'https://www.gutenberg.org/files/345/345-0.txt'        #Dracula
```

```
[ ] #first we will read both the books and print few initial lines
    T1_odyssey = urlopen(url1).read()
    T2_dracula = urlopen(url2).read()
```

```
[ ] T1_odyssey

    b'\xef\xbb\xbfThe Project Gutenberg eBook of The Odyssey, by Homer\r\n\r\nThis eBook is for the use of anyone anywhere in the United States and\r\nmost other parts of the world at no cost and with almost no
```

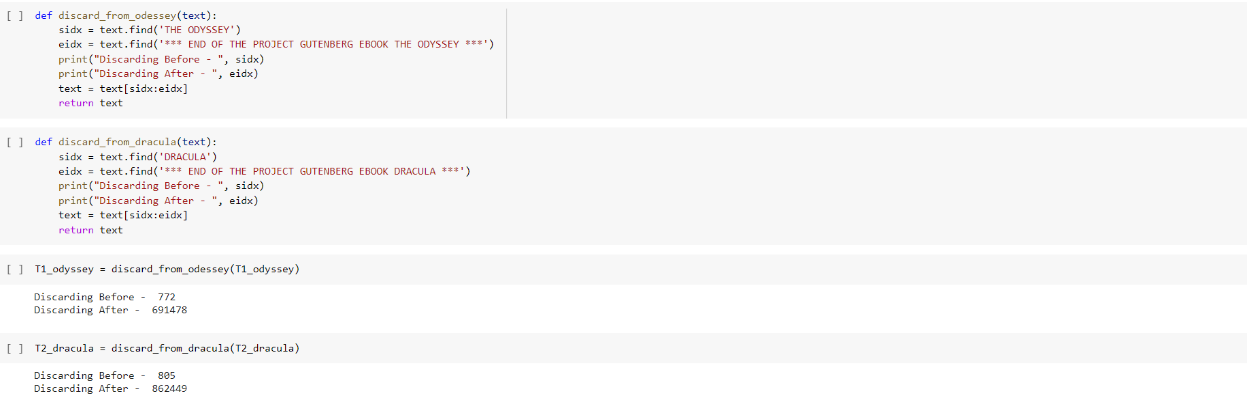
```
[ ] T2_dracula

    b'\xef\xbb\xbfThe Project Gutenberg eBook of Dracula, by Bram Stoker\r\n\r\nThis eBook is for the use of anyone anywhere in the United States and\r\nmost other parts of the world at no cost and with almost
```

```
[ ] T1_odyssey = T1_odyssey.decode('utf-8')
```

```
[ ] T2_dracula = T2_dracula.decode('utf-8')
```

Performing simple text-preprocessing steps and tokenizing both T1_odeyssey and T2_dracula

We are removing all the unnecessary text from the files.

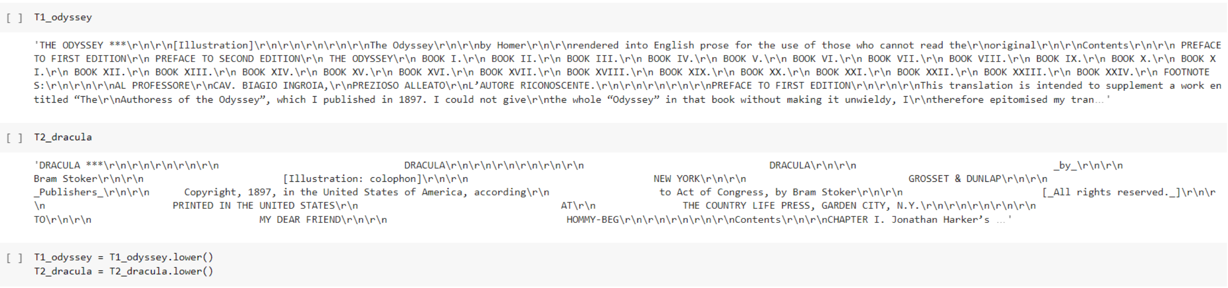


```
[ ] def discard_from_odessey(text):
        sidx = text.find('THE ODYSSEY')
        eidx = text.find('*** END OF THE PROJECT GUTENBERG EBOOK THE ODYSSEY ***')
        print("Discarding Before - ", sidx)
        print("Discarding After - ", eidx)
        text = text[sidx:eidx]
        return text
```

```
[ ] def discard_from_dracula(text):
        sidx = text.find('DRACULA')
        eidx = text.find('*** END OF THE PROJECT GUTENBERG EBOOK DRACULA ***')
        print("Discarding Before - ", sidx)
        print("Discarding After - ", eidx)
        text = text[sidx:eidx]
        return text
```

```
[ ] T1_odyssey = discard_from_odessey(T1_odyssey)

    Discarding Before -  772
    Discarding After -  691478
```

```
[ ] T2_dracula = discard_from_dracula(T2_dracula)

    Discarding Before -  805
    Discarding After -  862449
```

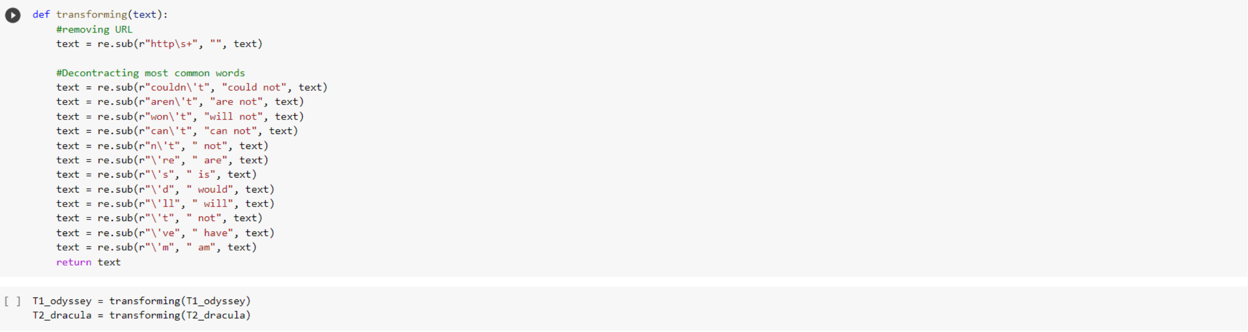


```
[ ] T1_odyssey

    'THE ODYSSEY ***\r\n\r\n[Illustration]\r\n\r\n\r\n\r\n\r\n\r\nThe Odyssey\r\n\r\nby Homer\r\n\r\nrendered into English prose for the use of those who cannot read the\r\noriginal\r\n\r\nContents\r\n\r\n PREFACE
    TO FIRST EDITION\r\n PREFACE TO SECOND EDITION\r\n THE ODYSSEY\r\n BOOK I.\r\n BOOK II.\r\n BOOK III.\r\n BOOK IV.\r\n BOOK V.\r\n BOOK VI.\r\n BOOK VII.\r\n BOOK VIII.\r\n BOOK IX.\r\n BOOK X.\r\n BOOK X
    I.\r\n BOOK XII.\r\n BOOK XIII.\r\n BOOK XIV.\r\n BOOK XV.\r\n BOOK XVI.\r\n BOOK XVII.\r\n BOOK XVIII.\r\n BOOK XIX.\r\n BOOK XX.\r\n BOOK XXI.\r\n BOOK XXII.\r\n BOOK XXIII.\r\n BOOK XXIV.\r\n FOOTNOTE
    S:\r\n\r\n\r\nAL PROFESSORE\r\nCAV. BIAGIO INGROIA,\r\nPREZIOSO ALLEATO\r\nL’AUTORE RICONOSCENTE.\r\n\r\n\r\n\r\n\r\n\r\nPREFACE TO FIRST EDITION\r\n\r\n\r\nThis translation is intended to supplement a work en
    titled “The\r\nAuthoress of the Odyssey”, which I published in 1897. I could not give\r\nthe whole “Odyssey” in that book without making it unwieldy, I\r\ntherefore epitomised my tran…'
```

```
[ ] T2_dracula

    'DRACULA ***\r\n\r\n\r\n\r\n\r\n                               DRACULA\r\n\r\n\r\n\r\n\r\n\r\n                              DRACULA\r\n\r\n                            _by_\r\n\r\n
    Bram Stoker\r\n\r\n                       [Illustration: colophon]\r\n\r\n                            NEW YORK\r\n\r\n                           GROSSET & DUNLAP\r\n\r\n
    _Publishers_\r\n\r\n      Copyright, 1897, in the United States of America, according\r\n                  to Act of Congress, by Bram Stoker\r\n\r\n                          [_All rights reserved._]\r\n\r
    \n                    PRINTED IN THE UNITED STATES\r\n                               AT\r\n                THE COUNTRY LIFE PRESS, GARDEN CITY, N.Y.\r\n\r\n\r\n\r\n\r\n
    TO\r\n\r\n                       MY DEAR FRIEND\r\n\r\n                                HOMMY-BEG\r\n\r\n\r\n\r\n\r\nContents\r\n\r\nCHAPTER I. Jonathan Harker’s …'
```

```
[ ] T1_odyssey = T1_odyssey.lower()
    T2_dracula = T2_dracula.lower()
```

Using regular expression to decontract certain words to standard 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
```

```
[ ] T1_odyssey = transforming(T1_odyssey)
    T2_dracula = transforming(T2_dracula)
```

Tokenizing both the book

```
[ ] def tokenizing_book(text):
        tokens = text.split()
        final_word_bag = []

        for word in tokens:
            if word.isdigit():
                converted_word = p.number_to_words(word)
                final_word_bag.append(converted_word)
            else:
                final_word_bag.append(word)

        return ' '.join(final_word_bag)
```

```
[ ] T1_odyssey = tokenizing_book(T1_odyssey)
    T2_dracula = tokenizing_book(T2_dracula)
```

```
[ ] T1_odyssey
```

```
'the odyssey *** [illustration] the odyssey by homer rendered into english prose for the use of those who cannot read the original contents preface to first edition preface to second edition the odyssey bo
ok i. book ii. book iii. book iv. book v. book vi. book vii. book viii. book ix. book x. book xi. book xii. book xiii. book xiv. book xv. book xvi. book xvii. book xviii. book xix. book xx. book xxi. book
xxii. book xxiii. book xxiv. footnotes: al professore cav. biagio ingroia, prezioso alleato l'autore riconoscente. preface to first edition this translation is intended to supplement a work entitled "the a
uthoress of the odyssey", which i published in 1897. i could not give the whole "odyssey" in that book without making it unwieldy, i therefore epitomised my translation, which was already completed and whi
ch i now publish in full. i shall not here argue the two main points dealt with in the work just mentioned; i have nothing either to add to, or to withdraw from, what i have there w…'
```

```
[ ] T2_dracula
```

```
'dracula *** dracula dracula _by_ bram stoker [illustration: colophon] new york grosset & dunlap _publishers_ copyright, 1897, in the united states of america, according to act of congress, by bram stoker
[_all rights reserved._] printed in the united states at the country life press, garden city, n.y. to my dear friend hommy-beg contents chapter i. jonathan harker's journal chapter ii. jonathan harker's jo
urnal chapter iii. jonathan harker's journal chapter iv. jonathan harker's journal chapter v. letters—lucy and mina chapter vi. mina murray's journal chapter vii. cutting from "the dailygraph," eight augus
t chapter viii. mina murray's journal chapter ix. mina murray's journal chapter x. mina murray's journal chapter xi. lucy westenra's diary chapter xii. dr. seward's diary chapter xiii. dr. seward's diary c
hapter xiv. mina harker's journal chapter xv. dr. seward's diary chapter xvi. dr. seward's diary chapter xvii. dr. seward's diary chapter xviii. dr. seward's diary chapter xix. jona…'
```

Removing non-alphabetic characters

```
[ ] def remove_non_alpha(text):
        tokens = text.split()
        final_word_bag = [word for word in tokens if word.isalpha()]
        return ' '.join(final_word_bag)
```

```
[ ] T1_odyssey = remove_non_alpha(T1_odyssey)
    T2_dracula = remove_non_alpha(T2_dracula)
```

Performing Lemmatization

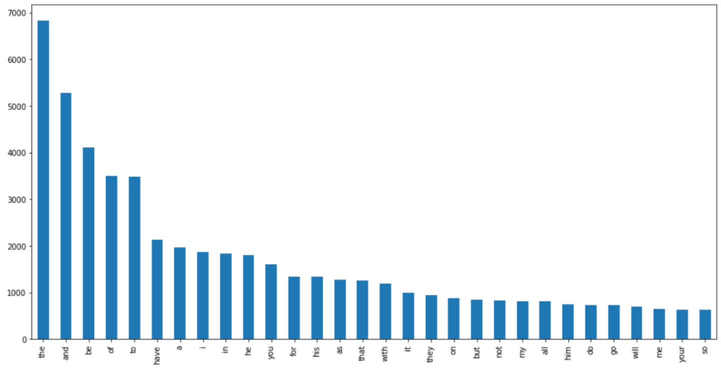
```
[ ] def lemmatize_word(text):
        word_tokens = text.split()
        lemmas = [lemmatizer.lemmatize(word, pos ='v') for word in word_tokens]
        return ' '.join(lemmas)
```

```
[ ] T1_odyssey = lemmatize_word(T1_odyssey)
    T2_dracula = lemmatize_word(T2_dracula)
```

Analysing the frequency distribution of tokens in T1_odessey and T2_dracula

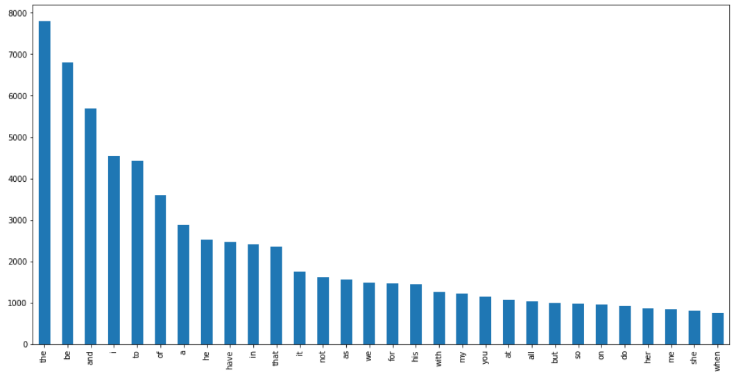
```
[ ] plt.figure(figsize=(16,8))
    tokens = word_tokenize(T1_odyssey)
    pd.Series(tokens).value_counts()[:30].plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a48b47fd0>
```



```
[ ]  plt.figure(figsize=(16,8))
     tokens = word_tokenize(T2_dracula)
     pd.Series(tokens).value_counts()[:30].plot(kind='bar')
```

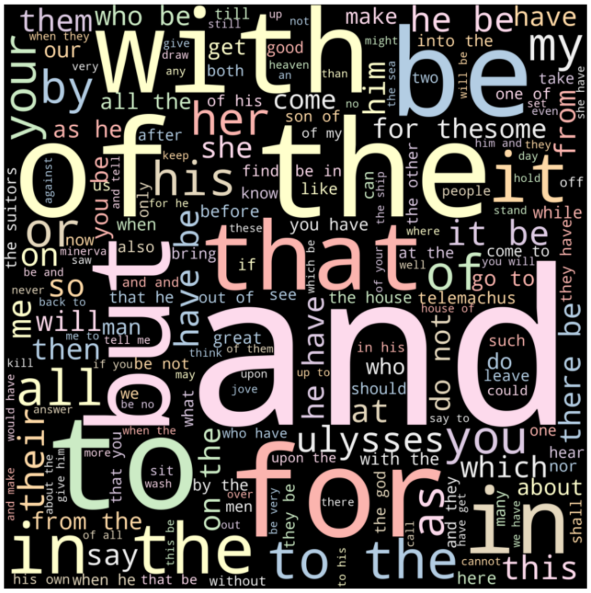
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a47cbd490>
```

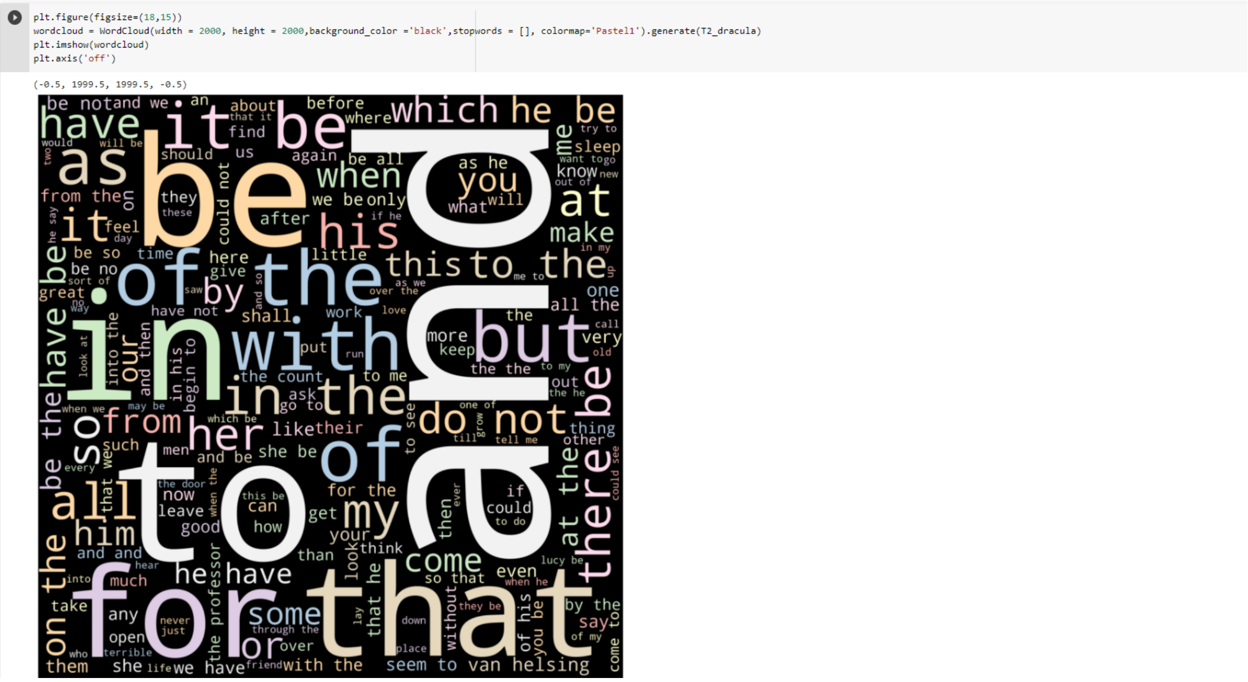


Generating Word Cloud

```
plt.figure(figsize=(15,15))
wordcloud = WordCloud(width = 2000, height = 2000,background_color ='black',stopwords = [], colormap='Pastel1').generate(T1_odyssey)
plt.imshow(wordcloud)
plt.axis('off')
```

```
(-0.5, 1999.5, 1999.5, -0.5)
```



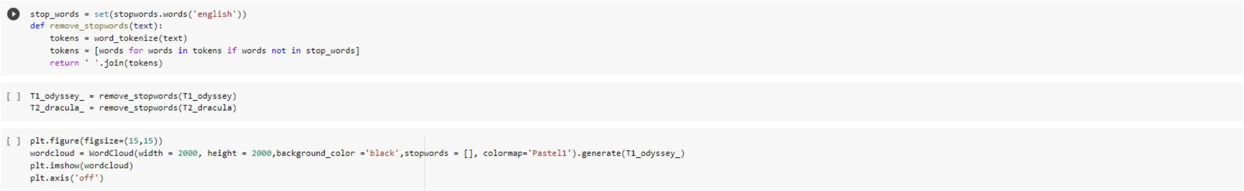


```
plt.figure(figsize=(18,15))
wordcloud = WordCloud(width = 2000, height = 2000,background_color ='black',stopwords = [], colormap='Pastel1').generate(T2_dracula)
plt.imshow(wordcloud)
plt.axis('off')
```

```
(-0.5, 1999.5, 1999.5, -0.5)
```

From the above word cloud visualisation, we can infer that the most frequently are mainly stopped words like ‘to’, ‘of’, ‘be’,’ the’. These words do not contribute to the meaning of the sentence. These stop words can easily be removed and the resulting in a better word cloud.

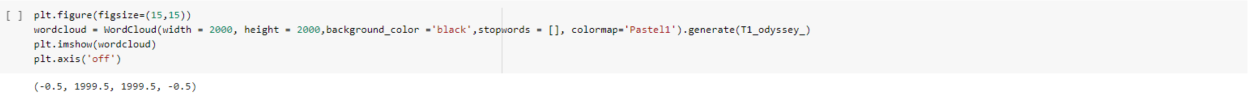
Removing the STOPWORDS and again generating Word Cloud to identify the potential differences between the word clouds before and after removing the STOPWORDS.



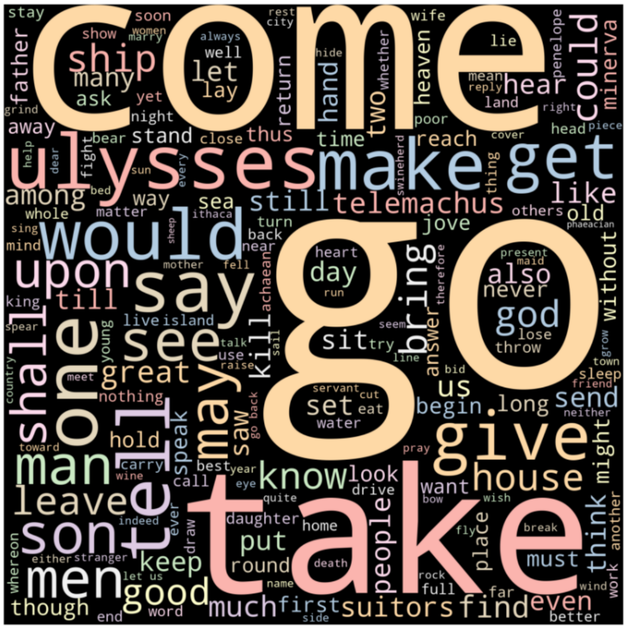
```
stop_words = set(stopwords.words('english'))
def remove_stopwords(text):
    tokens = word_tokenize(text)
    tokens = [words for words in tokens if words not in stop_words]
    return ' '.join(tokens)
```

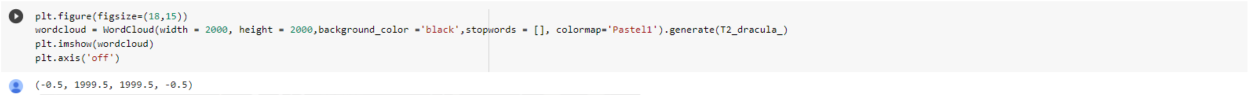
```
[ ] T1_odyssey_ = remove_stopwords(T1_odyssey)
    T2_dracula_ = remove_stopwords(T2_dracula)
```

```
[ ] plt.figure(figsize=(15,15))
    wordcloud = WordCloud(width = 2000, height = 2000,background_color ='black',stopwords = [], colormap='Pastel1').generate(T1_odyssey_)
    plt.imshow(wordcloud)
    plt.axis('off')
```

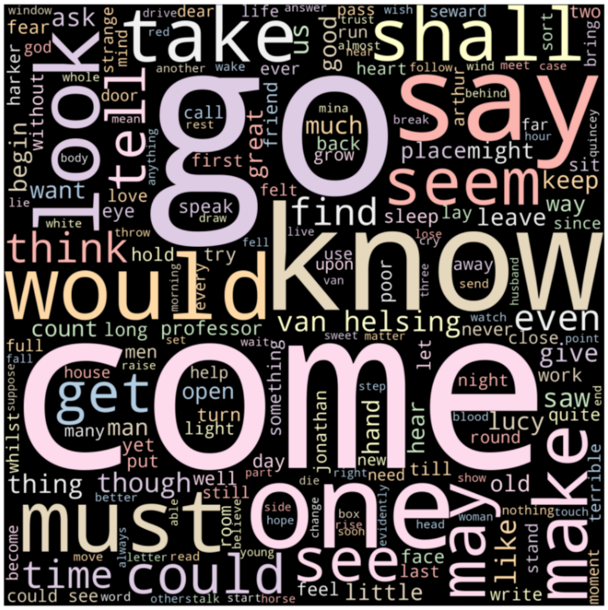


```
[ ]  plt.figure(figsize=(15,15))
     wordcloud = WordCloud(width = 2000, height = 2000,background_color ='black',stopwords = [], colormap='Pastel1').generate(T1_odyssey_)
     plt.imshow(wordcloud)
     plt.axis('off')

     (-0.5, 1999.5, 1999.5, -0.5)
```





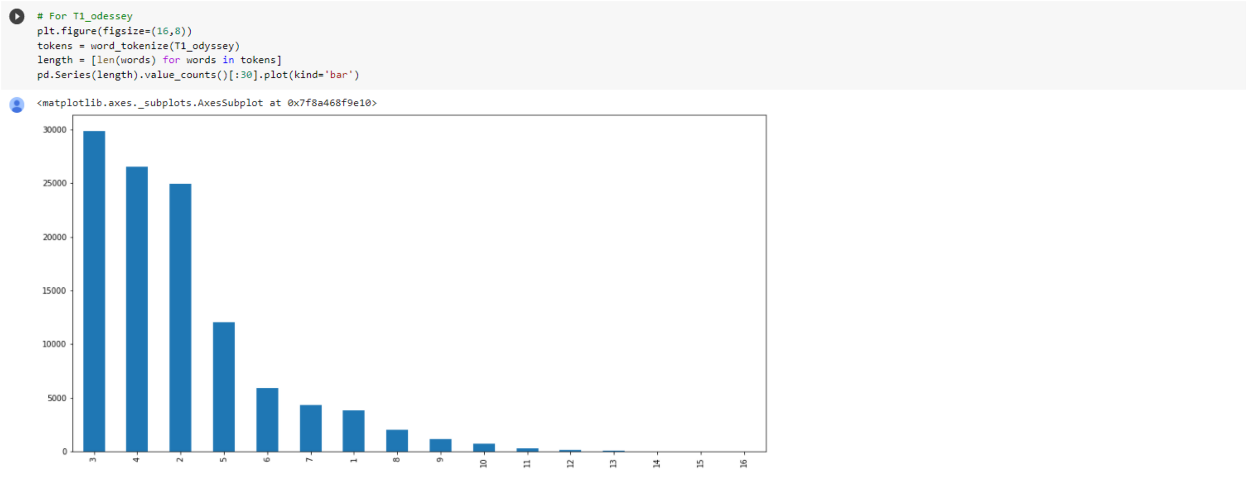
```
     plt.figure(figsize=(18,15))
     wordcloud = WordCloud(width = 2000, height = 2000,background_color ='black',stopwords = [], colormap='Pastel1').generate(T2_dracula_)
     plt.imshow(wordcloud)
     plt.axis('off')

     (-0.5, 1999.5, 1999.5, -0.5)
```



The words with no meaning are removed from the text after removing all the stop words.
Now more relevant words can be displayed with word clouds with words like ‘would’, ‘know’, ‘come’,’take’.

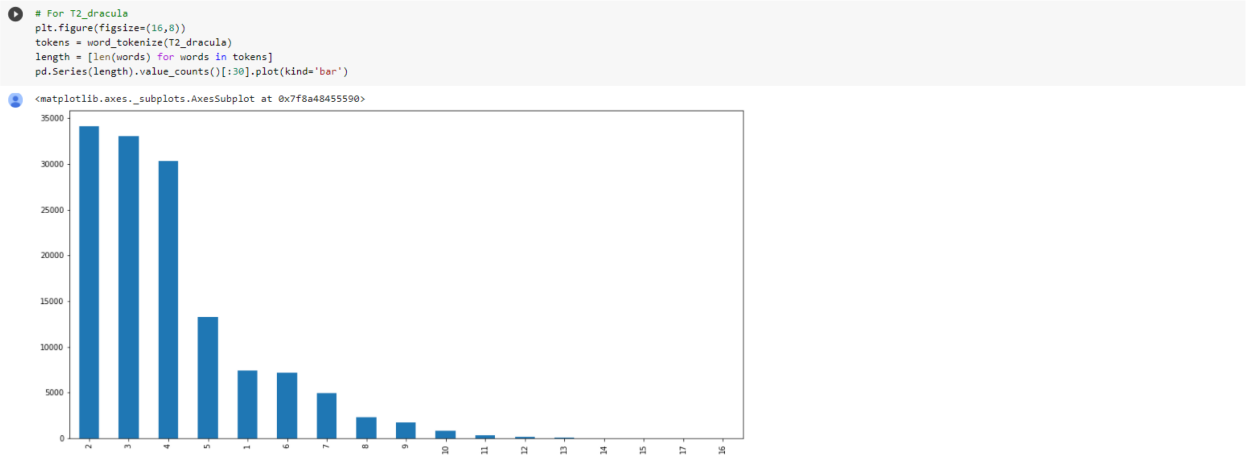
Evaluating the relationship between the word length and frequency for both T1_odessey and T2_dracula

Before removing StopWords



```
# For T1_odessey
plt.figure(figsize=(16,8))
tokens = word_tokenize(T1_odyssey)
length = [len(words) for words in tokens]
pd.Series(length).value_counts()[:30].plot(kind='bar')
```

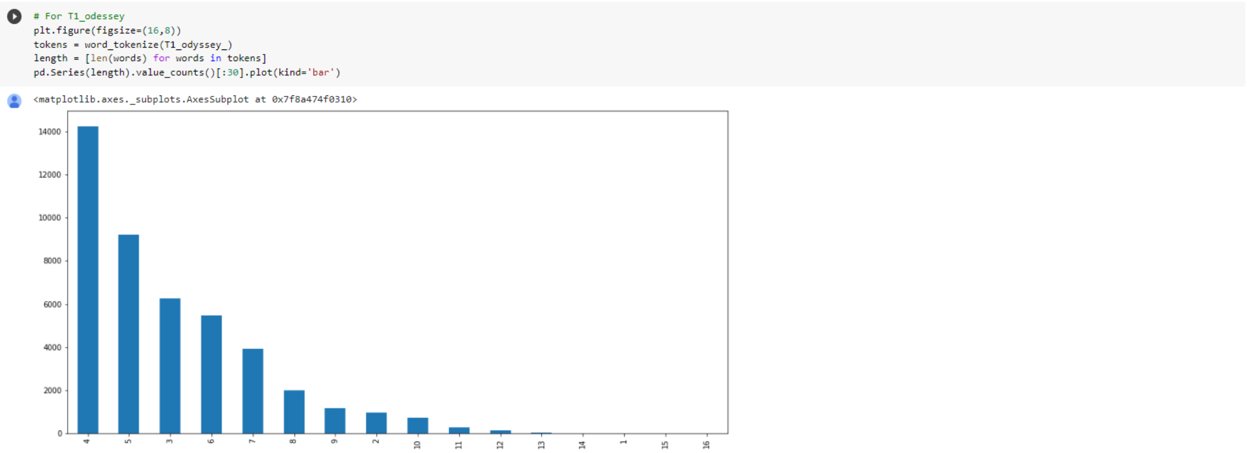
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a468f9e10>
```



```
# For T2_dracula
plt.figure(figsize=(16,8))
tokens = word_tokenize(T2_dracula)
length = [len(words) for words in tokens]
pd.Series(length).value_counts()[:30].plot(kind='bar')
```

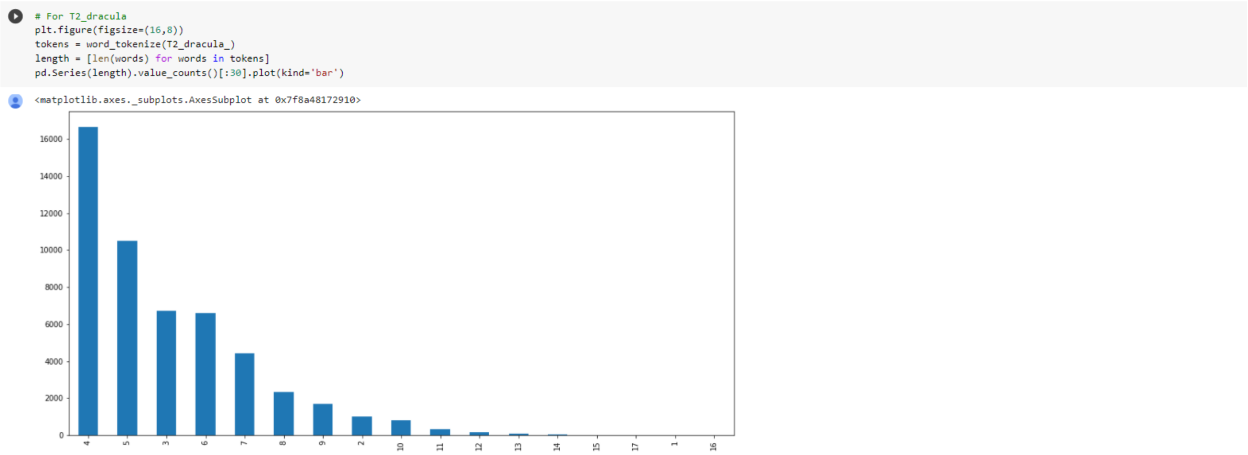
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a48455590>
```

After removing StopWords



```
# For T1_odessey
plt.figure(figsize=(16,8))
tokens = word_tokenize(T1_odyssey_)
length = [len(words) for words in tokens]
pd.Series(length).value_counts()[:30].plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a474f0310>
```

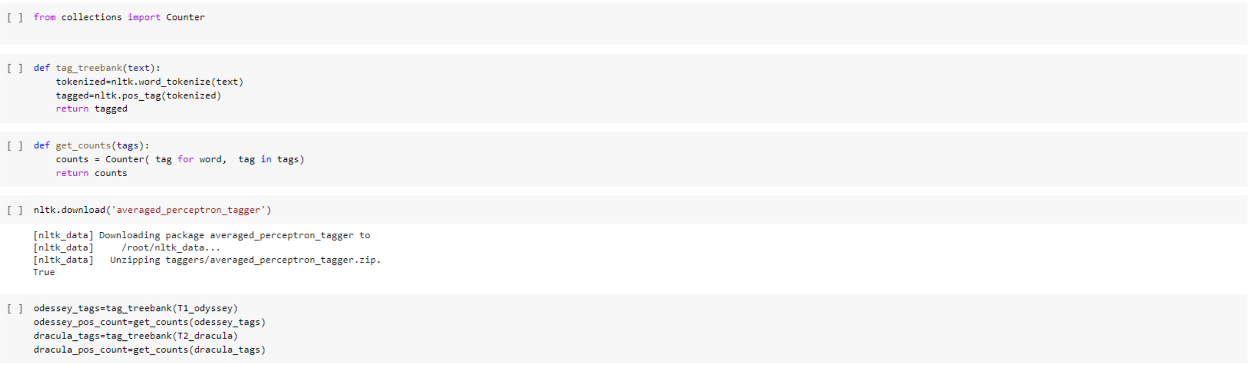


```
# For T2_dracula
plt.figure(figsize=(16,8))
tokens = word_tokenize(T2_dracula_)
length = [len(words) for words in tokens]
pd.Series(length).value_counts()[:30].plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a48172910>
```

The number of words of the length 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 been emerging, the stop words of length 3,4,5.

POS Tagging



```
[ ] from collections import Counter
```

```
[ ] def tag_treebank(text):
        tokenized=nltk.word_tokenize(text)
        tagged=nltk.pos_tag(tokenized)
        return tagged
```

```
[ ] def get_counts(tags):
        counts = Counter( tag for word,  tag in tags)
        return counts
```

```
[ ] nltk.download('averaged_perceptron_tagger')

    [nltk_data] Downloading package averaged_perceptron_tagger to
    [nltk_data]     /root/nltk_data...
    [nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
    True
```

```
[ ] odessey_tags=tag_treebank(T1_odyssey)
    odessey_pos_count=get_counts(odessey_tags)
    dracula_tags=tag_treebank(T2_dracula)
    dracula_pos_count=get_counts(dracula_tags)
```

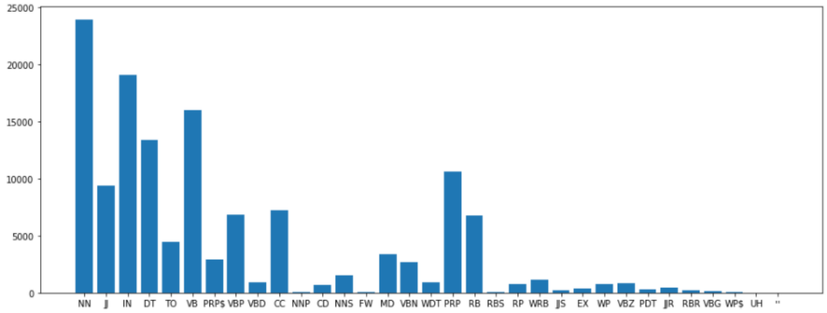
```
[ ] len(odessey_pos_count)

    32
```

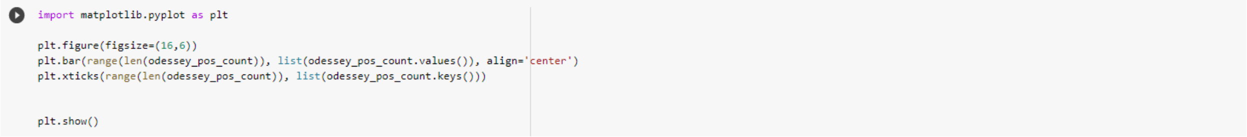
```
[ ] odessey_pos_count

    Counter({'$': 2,
             'CC': 6782,
             'CD': 774,
             'DT': 11036,
             'EX': 189,
             'FW': 31,
             'IN': 16283,
             'JJ': 7104,
             'JJR': 297,
             'JJS': 201,
             'MD': 2551,
             'NN': 20058,
             'NNP': 50,
             'NNS': 1344,
             'PDT': 248,
             'PRP': 8903,
             'PRP$': 2649,
             'RB': 5066,
             'RBR': 93,
             'RBS': 65,
             'RP': 668,
             'TO': 3479,
             'VB': 12382,
             'VBD': 755,
             'VBG': 204,
             'VBN': 2292,
             'VBP': 5173,
             'VBZ': 621,
             'WDT': 653,
             'WP': 959,
             'WP$': 43,
             'WRB': 981})
```

```
[ ] len(dracula_pos_count)

    33
```

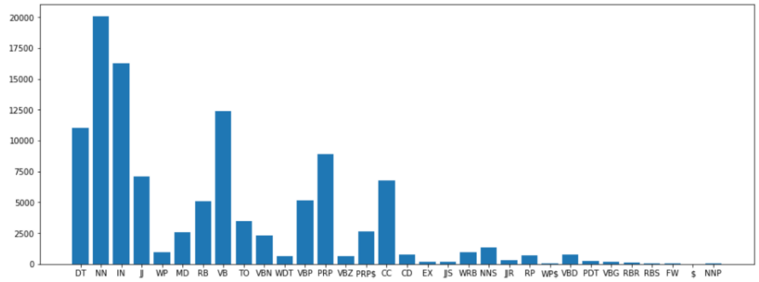
```
[ ] dracula_pos_count

    Counter({"''": 1,
             'CC': 7210,
             'CD': 650,
             'DT': 13401,
             'EX': 362,
             'FW': 50,
             'IN': 19083,
             'JJ': 9373,
             'JJR': 426,
             'JJS': 210,
             'MD': 3394,
             'NN': 23928,
             'NNP': 26,
             'NNS': 1504,
             'PDT': 310,
             'PRP': 10607,
             'PRP$': 2915,
             'RB': 6733,
             'RBR': 192,
             'RBS': 47,
             'RP': 776,
             'TO': 4429,
             'UH': 2,
             'VB': 16014,
             'VBD': 886,
             'VBG': 102,
             'VBN': 2685,
             'VBP': 6873,
             'VBZ': 795,
             'WDT': 894,
             'WP': 719,
             'WP$': 30,
             'WRB': 1133})
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(16,6))
plt.bar(range(len(dracula_pos_count)), list(dracula_pos_count.values()), align='center')
plt.xticks(range(len(dracula_pos_count)), list(dracula_pos_count.keys()))


plt.show()
```





```
import matplotlib.pyplot as plt

plt.figure(figsize=(16,6))
plt.bar(range(len(odessey_pos_count)), list(odessey_pos_count.values()), align='center')
plt.xticks(range(len(odessey_pos_count)), list(odessey_pos_count.keys()))


plt.show()
```



From this we can infer that the highest occurring tag is ‘NN’, and ‘Determinant’ Tags are on the lower frequency side. This is largely due to the removal of stopwords before POS Tagging.

CONCLUSION:-

We have learnt how to perform Text Processing, Tokenization and answer everything we asked of us using proper imports and libraries like pandas, matplotlib, NLTK.