NLP project Round1 report

Submitted by

Members of Team Language Revolution

Jatin Dahiya	19ucs033
Rishabh Sahu	19ucs129
Naman Dhanotia	19ucs128

Link to GitHub code repository

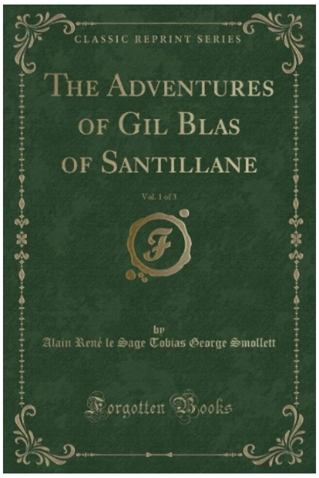
[Rishabhsahu325/NLP_Project_Round1: NLP project (github.com)]

Problem Description

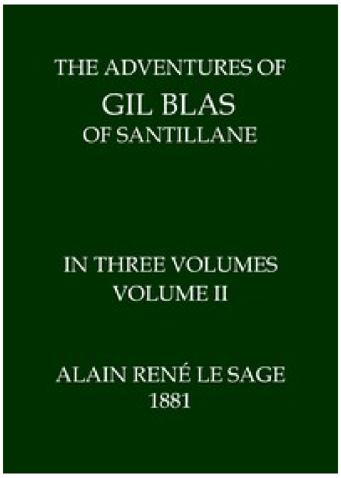
To take two books from http://gutenburg.org in .txt format and perform the following Natural Language processing operations on them

- Apply Data pre-processing on the text
- Generating frequency distributions of the words
- Creating word clouds from the text before and after removing stop words
- Evaluating relationship between word length and frequency
- Parts of Speech tagging for the words in the text

Books chosen for applying the processing


CLASSIC REPRINT SERIES
The Adventures of Gil Blas of Santillane
Vol. 1 of 3
by
Alain René le Sage Tobias George Smollett
Forgotten Books

T1: The Adventures of Gil Blas of Santillane, Volume I (of 3)



T2: The Adventures of Gil Blas of Santillane, Volume II (of 3)

Python Libraries/Modules used

Matplotlib : for drawing plots

Python re library (regular expressions library): For regular expressions

NumPy :for parameters of axes while plotting graphs

Nltk: :Used for tokenizing, removing stop words

Math :For calculating floor and ceil function values while plotting values

WordCloud :For creating word cloud

Collections :For getting the frequency mappings of the POS tags

Inferences after examining raw data

This raw text contains copyright related information, chapter headings, random blank lines and unprocessed text which cannot be directly processed.

Data Pre-processing and Preparation steps

We performed the following data pre-processing steps

1. Removing chapter number and chapter Headings
2. Removing all punctuation marks
3. Changing all text to lowercase
4. Converting short forms like can’t to actual representations
5. Tokenising the text into a list of words
6. Removing chapter headings and unrelated data
7. Removing hyperlinks

Preprocessing

In [5]:

```
# remove useless text from the data
def remove_useless_text(text):
    text = re.sub(r'\n_CHAPTER *[A-Z]*[A-Z]._|\nCHAPTER *[A-Z]*[A-Z].|\n_([\s\S]*?)_|\n([\s\S]*?)\*\*\* START OF THE PROJECT ([\s\S]*?)\*\*\*', ' ', te
    text = re.sub(r'\n\*\*\* END OF THE PROJECT ([\s\S]*)',' ',text)
    return text
```

In [6]:

```
# reading text from the book and told
with open('pg66677.txt', 'r',encoding='utf-8') as f:
    text_book1 = ''.join(f.readlines())

with open('pg66678.txt', 'r',encoding='utf-8') as f:
    text_book2 = ''.join(f.readlines())
```

In [7]:

```
text_book1 = remove_useless_text(text_book1)
text_book2= remove_useless_text(text_book2)
```

In [8]:

```
# converting all text to lower case and removing any link
def to_lower(text):
    text = text.lower()
    re.sub(r"http\S+", "", text)
    return text
```

In [9]:

```
text_book1 = to_lower(text_book1)
text_book2=to_lower(text_book2)
```

In [10]:

```
#converting short forms to full forms
def conversion(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
```

In [11]:

```
text_book1 = conversion(text_book1)
text_book2= conversion(text_book2)
```

In [12]:

```
# removing all the puntuations from the text
def remove_punctuations(text):
    text = re.sub(r"[^a-zA-Z0-9]", " ", text)
    return text
```

In [13]:

```
text_book1= remove_punctuations(text_book1)
text_book2=remove_punctuations(text_book2)
```

Tokenization

In [14]:

```
from nltk.tokenize import word_tokenize
nltk.download('punkt')

# splitting text into words
def tokonize_word(text):
    words = word_tokenize(text)
    return words
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

In [15]:

```
words_book1 = tokonize_word(text_book1)
words_book2 = tokonize_word(text_book2)
```

Frequency analysis

In [16]:

```
# analyzing the the frequency of words
data_analysis_book1 = nltk.FreqDist(words_book1)
data_analysis_book1.plot(25, cumulative=False)
data_analysis_book2 = nltk.FreqDist(words_book2)
data_analysis_book2.plot(25, cumulative=False)
```

Word Cloud

In [17]:

```
def listToString(s):
    str1 = " "
    return (str1.join(s))

# converting list to string
def list_to_string(words):
    text = listToString(words)
    return text
```

In [18]:

```
text_book1 = list_to_string(words_book1)
text_book2 = list_to_string(words_book2)
```

In [19]:

```
# creating word cloud for book 1
wc_1 = WordCloud(background_color="white", width=1000, height=1000, random_state=1,stopwords= [],collocations=False).generate(text_book1)
plt.imshow(wc_1)
```

Removing stop words

StopWords

In [21]:

```
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[21]:

```
True
```

In [22]:

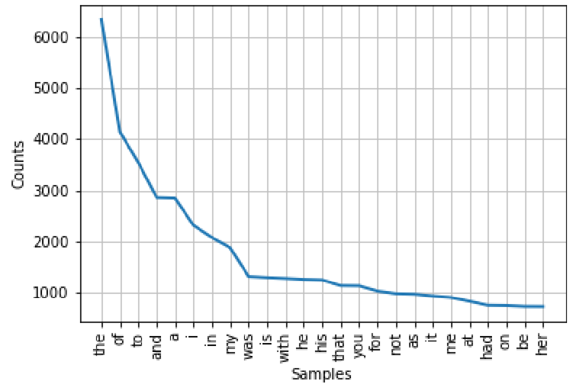
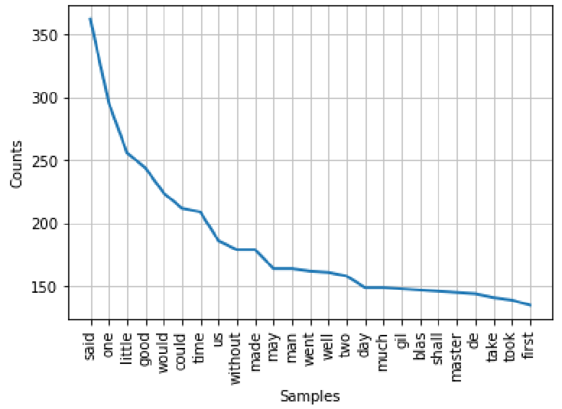
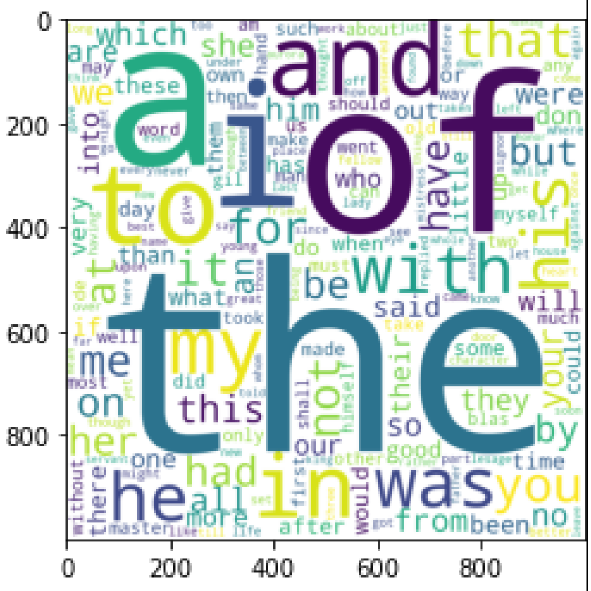
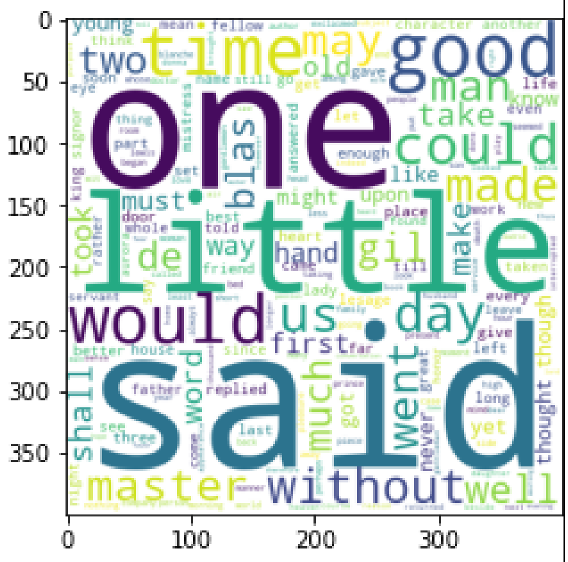
```
# removing the stopwords
def remove_stopwords(words):
    words = [w for w in words if w not in stopwords.words("english")]
    return words
```

In [23]:

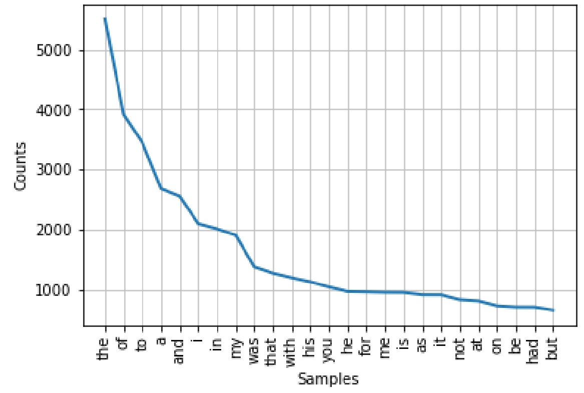
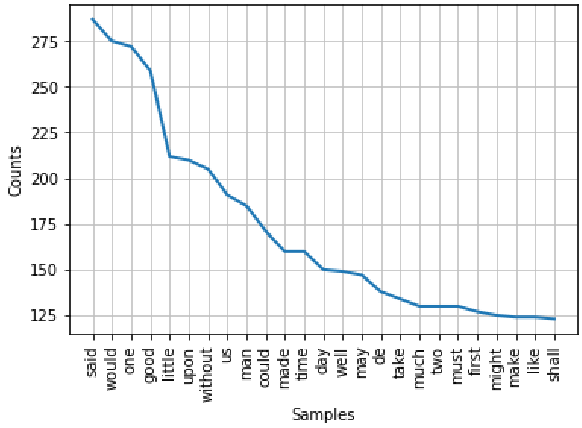
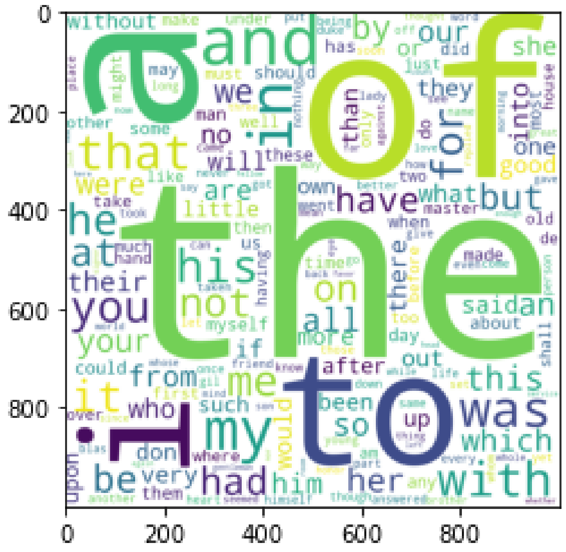
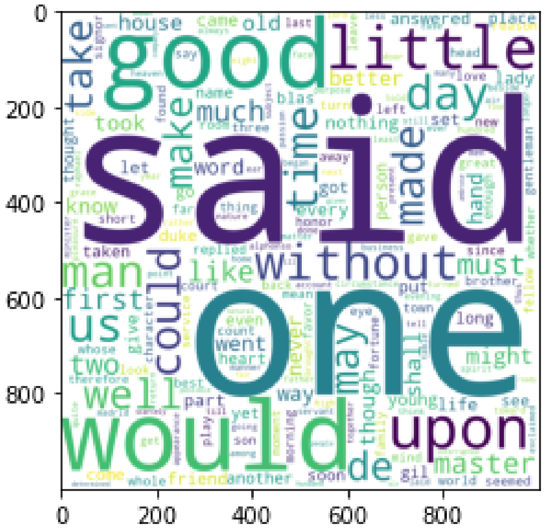
```
words_book1 = remove_stopwords(words_book1)
words_book2 =remove_stopwords(words_book2)
```

Illustrations (Word clouds and word wise frequency plots)

T1

Before	After
	
	

T2

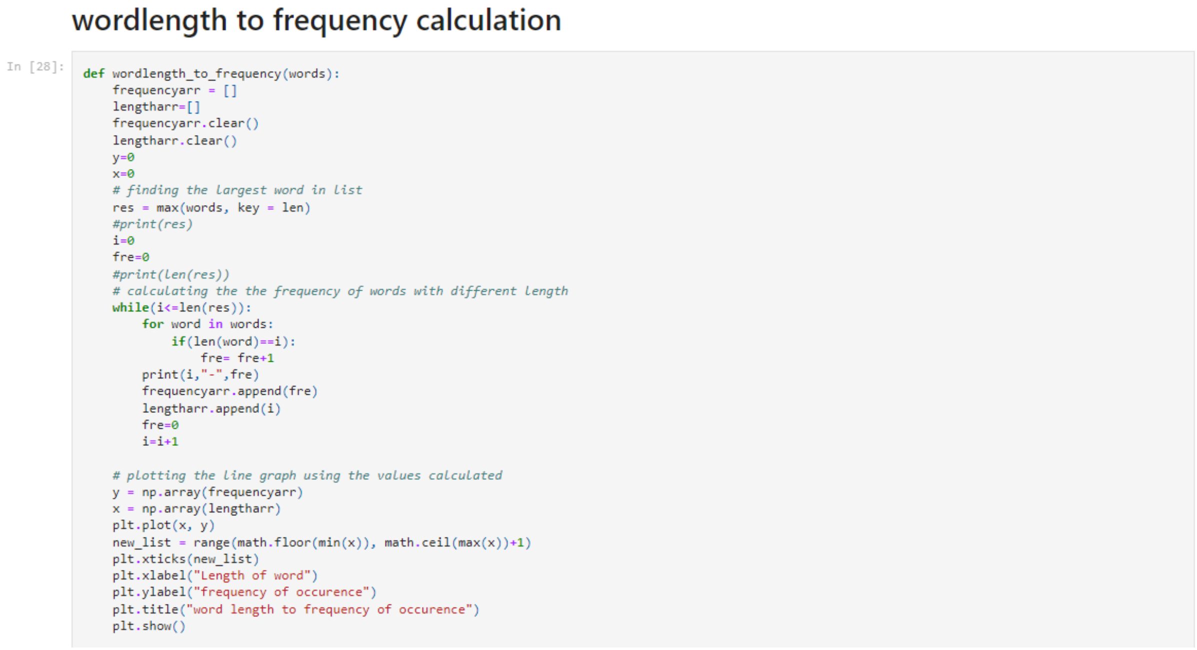
Before	After
 	
 	

Inference from word Clouds

- The word clouds before and after removing stop words are quite different due to the high frequency of many of these stop words. One of the reasons may be that stop words can be used in a variety of contexts whereas nouns and verbs are more restricted to the situations to which they relate to.
- After removing stop words, we are able to find the set of words which provide us meaning and context about the document.

Word length – frequency

Here we are calculating the word length and their frequency of occurrence.

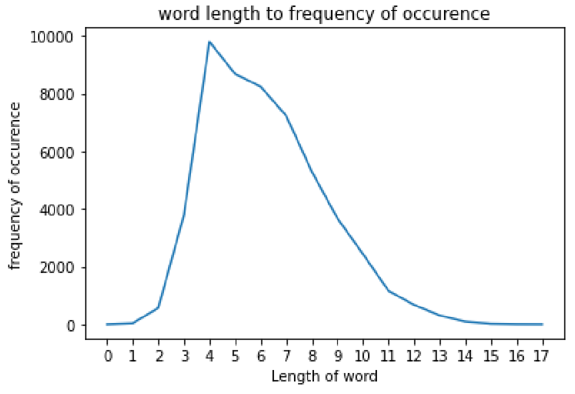
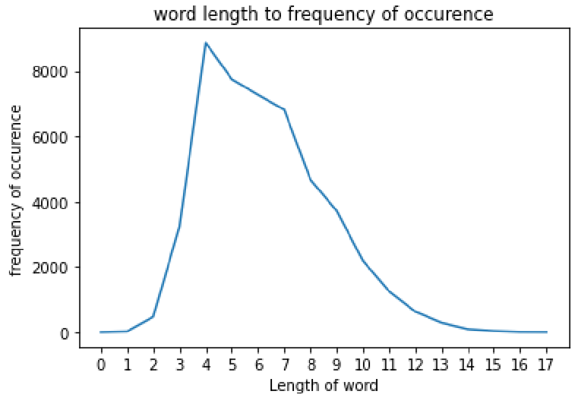


wordlength to frequency calculation

In [28]:

```
def wordlength_to_frequency(words):
    frequencyarr = []
    lengtharr=[]
    frequencyarr.clear()
    lengtharr.clear()
    y=0
    x=0
    # finding the largest word in list
    res = max(words, key = len)
    #print(res)
    i=0
    fre=0
    #print(len(res))
    # calculating the the frequency of words with different length
    while(i<=len(res)):
        for word in words:
            if(len(word)==i):
                fre= fre+1
        print(i,"-",fre)
        frequencyarr.append(fre)
        lengtharr.append(i)
        fre=0
        i=i+1

    # plotting the line graph using the values calculated
    y = np.array(frequencyarr)
    x = np.array(lengtharr)
    plt.plot(x, y)
    new_list = range(math.floor(min(x)), math.ceil(max(x))+1)
    plt.xticks(new_list)
    plt.xlabel("Length of word")
    plt.ylabel("frequency of occurence")
    plt.title("word length to frequency of occurence")
    plt.show()
```

Illustration: Word length – frequency plots

T1	T2
	

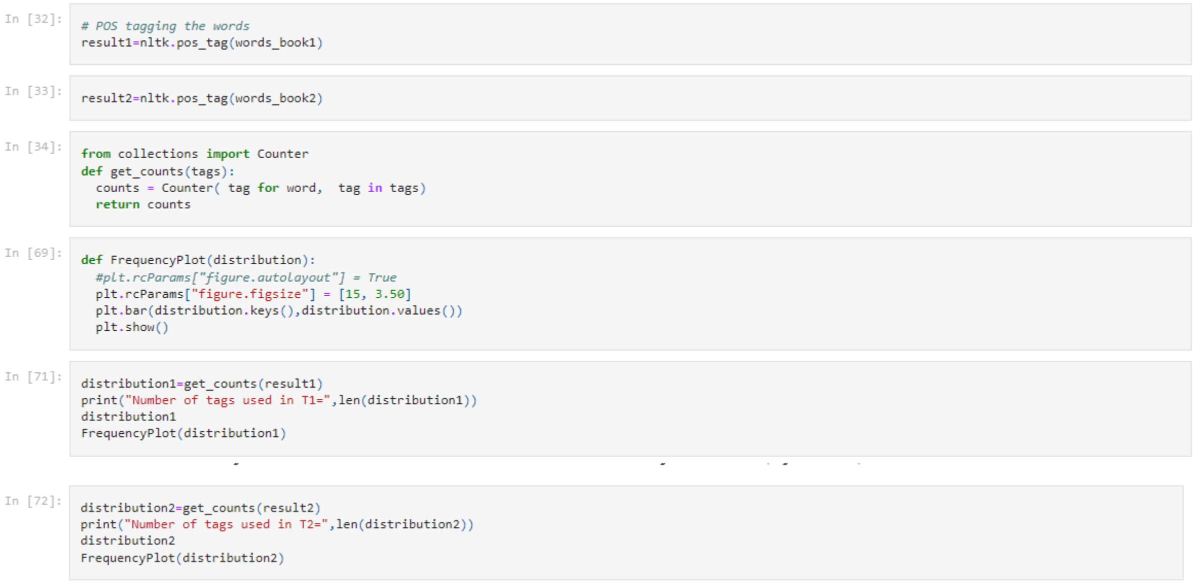
Inferences from word length- frequency plot

- For both the books Words having length between 3 to 5 are the most frequently occurring words in these books. After those words with larger lengths (up to a certain length) are frequent followed by words of length

1 to2. Very long words appear very rarely. Overall implying that most of the words lie in the length range of 3 to 5.

POS tagging

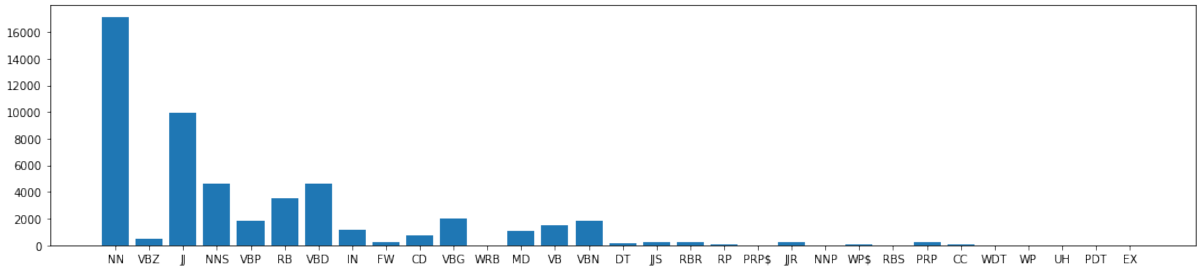
Here we are finding the tag associated with each word that was pre-processed.



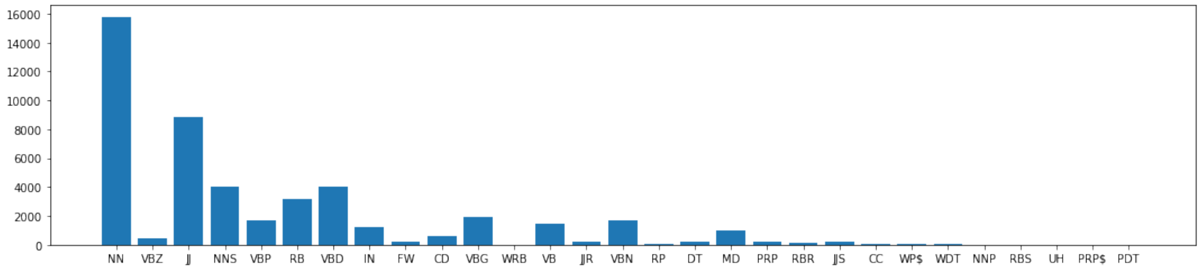
```
In [32]:  # POS tagging the words
          result1=nltk.pos_tag(words_book1)

In [33]:  result2=nltk.pos_tag(words_book2)

In [34]:  from collections import Counter
          def get_counts(tags):
            counts = Counter( tag for word,  tag in tags)
            return counts

In [69]:  def FrequencyPlot(distribution):
            #plt.rcParams["figure.autolayout"] = True
            plt.rcParams["figure.figsize"] = [15, 3.50]
            plt.bar(distribution.keys(),distribution.values())
            plt.show()

In [71]:  distribution1=get_counts(result1)
          print("Number of tags used in T1=",len(distribution1))
          distribution1
          FrequencyPlot(distribution1)

In [72]:  distribution2=get_counts(result2)
          print("Number of tags used in T2=",len(distribution2))
          distribution2
          FrequencyPlot(distribution2)
```

Illustrations POS tagging

Book T1



Book T2



Inferences POS_tagging

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 T1 the most frequently occurring POS Tag is 'NN' with count 17139 followed by 'JJ' having count 9972.

In T2 the most frequently occurring POS Tag is 'NN' with count 15804 followed by 'JJ ' having count 8871.

Conclusions

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