NLP PROJECT

REPORT

Team Name : **NLPB**

Members -

- Vatsal Agarwal(17ucs175)
- Aniket Poddar(17ucs022)
- Udit Bansal(17dcs015)

ROUND-1

REPORT ROUND 1 –

We have used book The Book Is "RURAL LIFE OF ENGLAND" by William Howitt

Meta data –

- Book Name - RURAL LIFE OF ENGLAND
- Author – William Howitt
- Number of Words – 244,984
- Number of Characters(with space) – 1,392,196
- Number of Characters(without space) – 1,157,215
- Language – English

1) **Preprocessing of Data –** Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. we have used regex like mentioned below-

    1) **maketrans(str1,str2,str3)** - used to construct transition table and to specify the list of characters that you want to be replaced in the whole string or the characters that need to be deleted from the string -Str1- characters that need to be replaced. -str2- characters with which the

characters need to be replaced -Str3- list of characters that needs to be deleted. -It return the transition table used by translate function.

Purpose - We have used this function so that we can remove all the punctuations in our file so that we can work with words and perform our task easily without any hindrance like punctuations and get our desired result.

2) **translate(table) -** used to translate transition table to string , where Table is required to perform translation and this function returns string after translation.

Purpose - We have used this to perform translation from table to our string it's like a follow through for maketrans() function so that we can obtain a string with removed punctuations.

```
In [4]: translator = str.maketrans("","",string.punctuation)
        T=T.translate(translator)
        print(T[:999])
```

3) **re.sub(pattern, rep , string)** - here this function checks the "pattern" or substring in the "string" and if found replaces It with another substring i.e. sub.

Purpose - we have used this for deleting unwanted and unnecessary details of our document like deleting table of content and the unnecessary words spaces and all the stuff.

In the picture below we have used re.sub() function to remove the unwanted preface and different content of table.

```
In [5]: T = re.sub("Project[\s\S]*EMBELLISHMENTS","",T)
        print(T[:999])
```

```
The


                                                    Page
  1 Vignette Summerhouse near Claremont            Title
  2 Old English Hall                                   1
  3 GrouseShooting in the Highlands                   29
  4 Oxen Ploughing                                    58
  5 A Garden Scene                                    67
  6 The Solitary House                               139
  7 Cattle in the Shade                              164
  8 Sir Roger de Coverley and the Gipsies            165
  9 Ladies personating Gipsies                       195
 10 Daleswomen going to a Shout                      221
 11 Old Dalesman and Traveller                       248
 12 Figures on a Screen in Annesley Hall             286
 13 The Otter Hunt by Bewick                         302
 14 Classical
```

In the picture below we have used re.sub() function to remove the Title and haedings in the book.

```
In [6]: T=re.sub("[A-Z]{2,}","",T)
        print(T[:999])
```

```
The


                                                    Page
  1 Vignette Summerhouse near Claremont            Title
  2 Old English Hall                                   1
  3 GrouseShooting in the Highlands                   29
  4 Oxen Ploughing                                    58
  5 A Garden Scene                                    67
  6 The Solitary House                               139
  7 Cattle in the Shade                              164
  8 Sir Roger de Coverley and the Gipsies            165
  9 Ladies personating Gipsies                       195
 10 Daleswomen going to a Shout                      221
 11 Old Dalesman and Traveller                       248
 12 Figures on a Screen in Annesley Hall             286
 13 The Otter Hunt by Bewick                         302
 14 Classical
```

In the picture below we have used re.sub() function to remove the numbers from the book.

```
In [7]: T=re.sub("[0-9]+","",T)
        print(T[:999])
```

```
The


                                                    Page
  Vignette Summerhouse near Claremont              Title
  Old English Hall
  GrouseShooting in the Highlands
  Oxen Ploughing
  A Garden Scene
  The Solitary House
  Cattle in the Shade
  Sir Roger de Coverley and the Gipsies
  Ladies personating Gipsies
 Daleswomen going to a Shout
 Old Dalesman and Traveller
 Figures on a Screen in Annesley Hall
 The Otter Hunt by Bewick
 Classical Rural Scenes
```

4) **lower() –** used to convert a string to lower case.

Purpose – to convert string to lower case ease of computation.

In the picture below we have used re.sub() function to convert string to lower case.

```
In [8]: T = T.lower()
        print(T[:999])
```

```
the


                                                                  page
    vignette summerhouse near claremont                    title
    old english hall
    grouseshooting in the highlands
    oxen ploughing
    a garden scene
    the solitary house
    cattle in the shade
    sir roger de coverley and the gipsies
    ladies personating gipsies
   daleswomen going to a shout
   old dalesman and traveller
   figures on a screen in annesley hall
   the otter hunt by bewick
   classical rural scenes
```

5) **split() -** is used to break the string specified separator .

Purpose - Here we are using it to break string into words.

6) **join() -** returns a string concatenated with the elements of iterable.

Purpose - Here we use it to make braked words into a single string but separated with space.

```
In [9]: T = " ".join(T.split())
        wordcloudT=T
        print(T[:999])
```

```
the page vignette summerhouse near claremont title old english hall grouseshooting in the highlands oxen ploughing a garden sce
ne the solitary house cattle in the shade sir roger de coverley and the gipsies ladies personating gipsies daleswomen going to
a shout old dalesman and traveller figures on a screen in annesley hall the otter hunt by bewick classical rural scenes scene i
n a town street by bewick wild horses in new forest purkiss's cottage new forest charcoalburners' hut wild english cattle in ch
illingham park woman driving geese procession of village maidens at whitsuntide morgan lewis shewing the last haunt of the fair
ies the village inn a sea scene a donkey race birdcatching tickling trout i i page preeminence of england as a place of country
residence its political and moral position the conveniences conferred by the perfection of the arts on social life its literatu
re spirit of freedom religious feeling and philanthropic institutions the delightfulness of its country resi
```

7) **word_tokenize(String) –** is used to extract token from string of characters.

Purpose - NOW WE HAVE TO FORMS TOKENS TO PERFORM FUTHER STEPS, Here we have used it to tokenize our previously formed string.

```
In [10]: from nltk.tokenize import word_tokenize
         T = word_tokenize(T)
         print(T[:999])
```

```
['\ufeffthe', 'page', 'vignette', 'summerhouse', 'near', 'claremont', 'title', 'old', 'english', 'hall', 'grouseshooting',
'in', 'the', 'highlands', 'oxen', 'ploughing', 'a', 'garden', 'scene', 'the', 'solitary', 'house', 'cattle', 'in', 'the', 's
hade', 'sir', 'roger', 'de', 'coverley', 'and', 'the', 'gipsies', 'ladies', 'personating', 'gipsies', 'daleswomen', 'going',
'to', 'a', 'shout', 'old', 'dalesman', 'and', 'traveller', 'figures', 'on', 'a', 'screen', 'in', 'annesley', 'hall', 'the',
'otter', 'hunt', 'by', 'bewick', 'classical', 'rural', 'scenes', 'scene', 'in', 'a', 'town', 'street', 'by', 'bewick', 'wil
d', 'horses', 'in', 'new', 'forest', 'purkiss', "'", 's', 'cottage', 'new', 'forest', 'charcoalburners', "'", 'hut', 'wild',
'english', 'cattle', 'in', 'chillingham', 'park', 'woman', 'driving', 'geese', 'procession', 'of', 'village', 'maidens', 'a
t', 'whitsuntide', 'morgan', 'lewis', 'shewing', 'the', 'last', 'haunt', 'of', 'the', 'fairies', 'the', 'village', 'inn',
'a', 'sea', 'scene', 'a', 'donkey', 'race', 'birdcatching', 'tickling', 'trout', 'i', 'i', 'page', 'preeminence', 'of', 'eng
land', 'as', 'a', 'place', 'of', 'country', 'residence', 'its', 'political', 'and', 'moral', 'position', 'the', 'convenience
s', 'conferred', 'by', 'the', 'perfection', 'of', 'the', 'arts', 'on', 'social', 'life', 'its', 'literature', 'spirit', 'o
f', 'freedom', 'religious', 'feeling', 'and', 'philanthropic', 'institutions', 'the', 'delightfulness', 'of', 'its', 'countr
y', 'residences', 'with', 'its', 'parks', 'lawns', 'woods', 'gardens', 'etc', 'the', 'variety', 'of', 'scenery', 'in', 'a',
'small', 'compass', 'advantages', 'of', 'its', 'climate', 'notwithstanding', 'all', 'just', 'cause', 'of', 'complaint', 'it
s', 'soil', 'sanctified', 'by', 'noble', 'deeds', 'and', 'intellectual', 'renown', 'real', 'superiority', 'of', 'england',
'as', 'a', 'place', 'of', 'residence', 'shewn', 'by', 'its', 'effects', 'on', 'foreigners', 'willis', "'", 's', 'descriptio
n', 'of', 'its', 'effect', 'on', 'him', 'enviable', 'position', 'of', 'the', 'english', 'country', 'gentleman', 'as', 'regar
ds', 'all', 'the', 'pleasures', 'and', 'advantages', 'of', 'life', 'every', 'art', 'and', 'energy', 'exerted', 'in', 'his',
'favour', 'by', 'them', 'his', 'house', 'surrounded', 'with', 'delights', 'the', 'news', 'and', 'the', 'luxuries', 'of', 'th
e', 'world', 'brought', 'to', 'his', 'table', 'books', 'music', 'paintings', 'at', 'his', 'command', 'farming', 'gardening',
'planting', 'fieldsports', 'all', 'within', 'his', 'grasp', 'scenes', 'which', 'offer', 'themselves', 'to', 'extend', 'his'
```

Till here we have performed all the necessary preprocessing steps so now we can go ahead and do the required task assign in the project.

2) Data Analysis –

We will first analyze the data using:

a) **wordcloud** which will tell us which word are in great frequency and by how much using the figure size in the plotted wordcloud,
b) **Graphs** using appropriate graphs to plot realties.

1) wordcloud –

wordcloud can be considered as a picture that contains most of the words that are available in the string but the size of the words depends on the number of times they appear in the string.

WordCloud() is a function that is used to generate a word cloud with the input values that are given through the available parameters like height, width ,size ,color etc. We have used variable wordcloud to store a OUTPUT of the function WordCloud().

```
In [16]: wordcloud = WordCloud(width = 800, height=800,
                               background_color='white',
                               min_font_size=10).generate(wordcloudT)

         plt.figure(figsize=(15,15),facecolor=None)
         plt.imshow(wordcloud)
         plt.axis("off")

         plt.show()
```

Which gives us following figure –

Result – this shows that words like great, country, one , time are of highest frequency in our book.

Now we want to have a wordcloud without the stopwords.

Stopwords - are words that are commonly used and because of that search engine are programmed to ignore that ex- a, an, the. -stopwords.words('English') is used to set the stopwords criteria to English language so that English language stopwords can be ignored.

Lemmatization - Lemmatization is the process of grouping together the different inflected forms of a word, we have used it here so that we can simplify our analysis.

```
In [11]: lemmatizer = WordNetLemmatizer()
         stop_words = set(stopwords.words('english'))

         lemmatized_T=[]

         for word in T:
             if len(word) >= 2 and word not in stop_words:
                 lemmatized_T.append(lemmatizer.lemmatize(word))

         print(lemmatized_T)
```

Code explained -

- Here WordNetLemmatizer() function is used to lemmatize our string

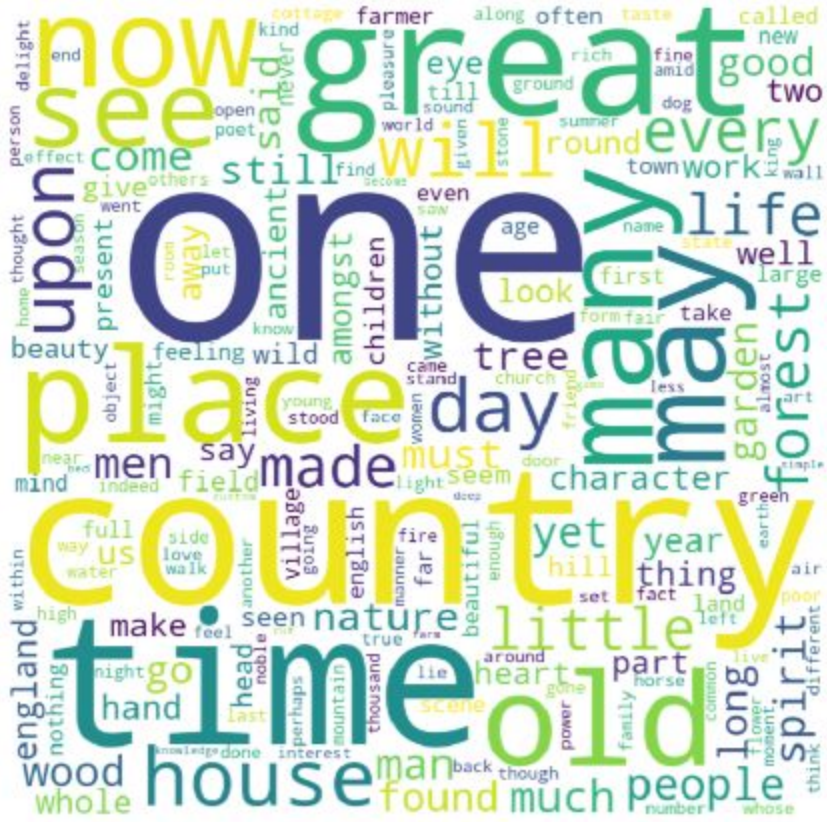
-Here the "for" loop we will be using will be for taking words into consideration whose length are 2 or greater than 2 as per our requirement.

- len(word) is used for calculating length of any word ith the string stop_word is a variable that is pre-calculated and stores the value of English stopwords.

Now we can generate the wordcloud without using stopwords as they are being excluded using function parameter as required in the question.

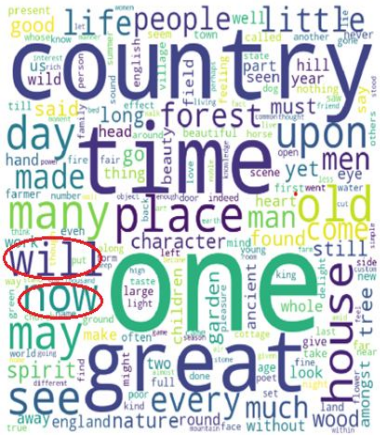
```
In [17]: wordcloud = WordCloud(width = 800, height=800,
                              background_color='white',
                               stopwords = set(STOPWORDS),
                              min_font_size=10).generate(wordcloudT)

         plt.figure(figsize=(15,15),facecolor=None)
         plt.imshow(wordcloud)
         plt.axis("off")

         plt.show()
```

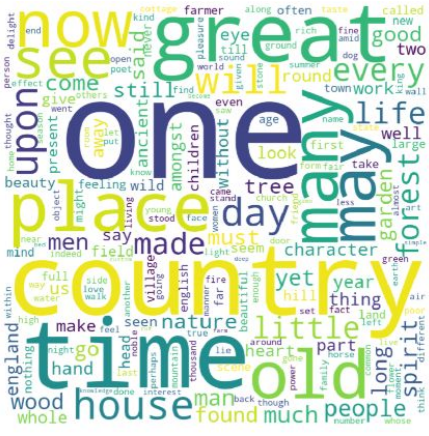
Here we have used stopwords = set(STOPWORDS) to build wordcloud consisting of no stopwords and the output is as –



Result – once again we can see that words like one, country are more in frequency.

Difference – Words like will, now were missing in the second wordcloud because they were stopwords but there e=were many which we are not able to see but change in wordcloud does signifies it.





Graphs – (We will be using Bar and Scatter Graph for plotting because it were giving us good result during our test.**)** We were asked to evaluate relationship between word length and frequency, to do this we have to keep track of word of a respective length and their frequency. To do this we have done following steps-

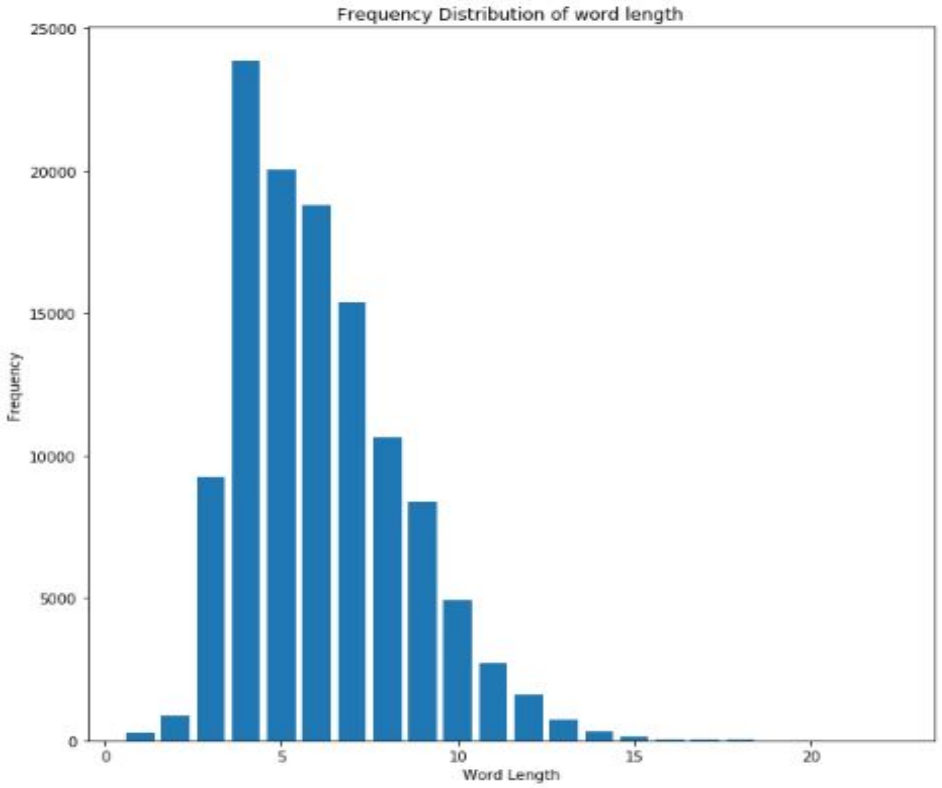
- initialize a list/array with zero value in it
- run a for loop through our string.
- if a new word with word length i.e not found in our list/array, store it in our list/array with initial value 1
- if it is increase its frequency by 1. Now we have a list with words and their frequency all we need to do is to plot it using plot function in PYTHON.

```
In [22]: len_list={}

         for i in range(len(lemmatized_T)):

             if len(lemmatized_T[i]) not in len_list:
                 len_list[len(lemmatized_T[i])] = 1

             else:
                 len_list[len(lemmatized_T[i])] += 1

         keys = list(len_list.keys())
         values = list(len_list.values())

         plt.figure(figsize=(10,10))
         plt.bar(keys,values)
         plt.xlabel("Word Length")
         plt.ylabel("Frequency")
         plt.title("Frequency Distribution of word length")
         plt.show()
```

Here-

- list is - len_list
- string- lemmatized_T
- X-axis-keys-"Word Length"
- Y-axis-values-"Frequency"

NOTE-plt.title is used to title the plot.



CONCLUSION OF THE ABOVE GRAPH - This shows that words mostly are of shorter length i.e around 3-10 rather than words which have length like 10 o . We can also conclude that our wordcloud that we constructed because words like one, country and **many big sized words** were abundant and from graph we can see that words **around length 5 and 6** are maximum.

Now to plot graph between "POS_TAGS" and their respective "Frequency", we have to keep track of POS_TAGS of words instead of word (like in previous question) and their frequency. To do this we have done following steps-

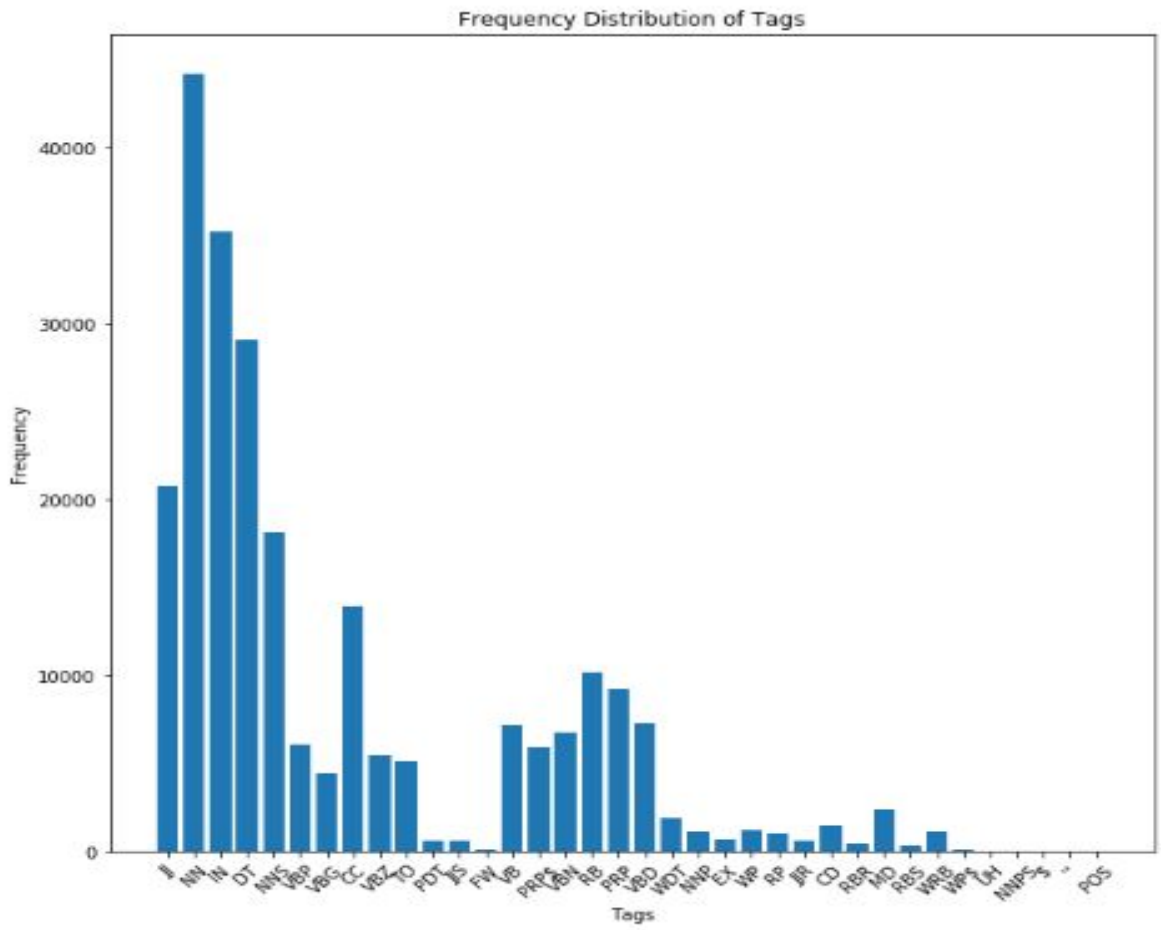
- initialize a list/array with zero value in it
- run a for loop through our string(POS_TAGGED) which is converted into pos_tagged string using - nltk.pos_tag(value).
- if a new tag with tag length is not found in our list/array, store it in our list/array with initial value 1
- if it is increase its frequency by 1. Now we have an list with words and their frequency all we need to do is to plot it using plot function in PYTHON.

Here is the code for the above steps –

```
In [21]: PosTags = nltk.pos_tag(T)

         tag_list={}

         for i in range(len(PosTags)):

             if PosTags[i][1] not in tag_list:
                 tag_list[PosTags[i][1]] = 1

             else:
                 tag_list[PosTags[i][1]] += 1

         keys = list(tag_list.keys())
         values = list(tag_list.values())

         plt.figure(figsize=(10,10))
         plt.bar(keys,values)
         plt.xlabel("Tags")
         plt.xticks(rotation = 45)
         plt.ylabel("Frequency")
         plt.title("Frequency Distribution of Tags")
         plt.show()
```

Here,

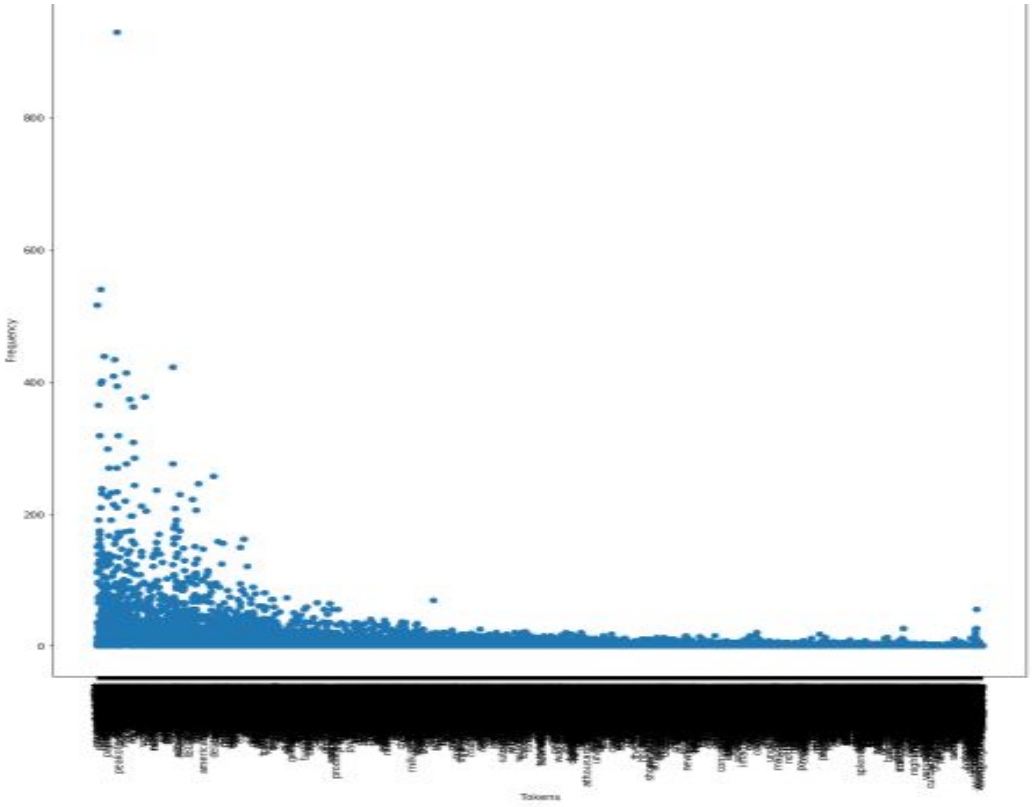
- STRING(POS_TAGGED_LIST)
- PosTags, which is obtained by using function "nltk.pos_tag(value)" where, value in function i the string we want to be TAGGED.



CONCLUSION OF THE ABOVE GRAPH - The graph above shows the frequency of tags , i.e, how many times a tag has been used. According to above graph in this book Nominal Noun is used the most while possessive ending(POS),Wh-adverb(WRB),UH ,NNPS ,etc were used the least. The initials are universal that got its name from "PENTREE BANK P.O.S TAGS".

Now to plot graph between "tokens" and their respective "frequency"- to do this we use function FreqDist which is a frequency distribution that can be defined as a function mapping from each sample to the number of times that sample occurred as an outcome. So here we have mapped the occurrence of each word in the book with the frequency of its occurrence.

```
In [20]: freq_dist = nltk.FreqDist(lemmatized_T)

keys = list(freq_dist.keys())
values = list(freq_dist.values())
labels=keys
plt.figure(figsize=(15,15))
plt.scatter(keys,values)
plt.xlabel("Tokens")
plt.ylabel("Frequency")
plt.xticks(keys, labels, rotation='vertical')
plt.title("Frequency Distribution of tokens")
plt.show()
```



Here the frequency of each words are plotted ,i.e, how many times each word has occurred. But it has an issue that is due to excess number of words the x-axis labels(i.e tokens) are co-incided. And we can also see that one is max which will be one and followed with country and all.

Round 2

REPORT ROUND 2

Meta data (BOOK 1)–

- Book Name - **RURAL LIFE OF ENGLAND**
- Author – William Howitt
- Number of Words – 244,984
- Number of Characters(with space) – 1,392,196
- Number of Characters(without space) – 1,157,215
- Language – English

Meta Data (BOOK 2) –

- Book Name – **The book of the thousand nights and a night (volume 3)**
- Author – Richard F. Burton
- Number of words – 130,683
- Number of characters (with space) – 714,758
- Number of characters(without space) – 589,082
- Language - English

Preprocessing of Data :

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. We have used regex in both books preprocessing steps were same in both the books which are like like mentioned below-

1. **maketrans(str1,str2,str3)** - used to construct transition table and to specify the list of characters that you want to be replaced in the whole string or the characters that need to be deleted from the string -Str1- characters that need to be replaced. -str2- characters with which the characters need to be replaced -Str3- list of characters that needs to be deleted. -It return the transition table used by translate function.

Purpose - We have used this function so that we can remove all the punctuations in our file so that we can work with words and perform our task easily without any hindrance like punctuations and get our desired result.

2. **translate(table) -** used to translate transition table to string , where Table is required to perform translation and this function returns string after translation.

   Purpose - We have used this to perform translation from table to our string it's like a follow through for maketrans() function so that we can obtain a string with removed punctuations.

```
In [4]: translator = str.maketrans("","",string.punctuation)
        T=T.translate(translator)
        print(T[:999])
```

3. **re.sub(pattern, rep , string)** - here this function checks the "pattern" or substring in the "string" and if found replaces It with another substring i.e. sub.

   Purpose - we have used this for deleting unwanted and unnecessary details of our document like deleting table of content and the unnecessary words spaces and all the stuff.

In the picture below we have used re.sub() function to remove the unwanted preface and different content of table.

```
In [5]: T = re.sub("Project[\s\S]*EMBELLISHMENTS","",T)
        print(T[:999])
```

```
The


                                                    Page
  1 Vignette Summerhouse near Claremont           Title
  2 Old English Hall                                  1
  3 GrouseShooting in the Highlands                  29
  4 Oxen Ploughing                                   58
  5 A Garden Scene                                   67
  6 The Solitary House                              139
  7 Cattle in the Shade                             164
  8 Sir Roger de Coverley and the Gipsies           165
  9 Ladies personating Gipsies                      195
 10 Daleswomen going to a Shout                     221
 11 Old Dalesman and Traveller                      248
 12 Figures on a Screen in Annesley Hall            286
 13 The Otter Hunt by Bewick                        302
 14 Classical
```

In the picture below we have used re.sub() function to remove the Title and haedings in the book.

```
In [6]: T=re.sub("[A-Z]{2,}","",T)
        print(T[:999])
```

```
The


                                                    Page
  1 Vignette Summerhouse near Claremont           Title
  2 Old English Hall                                  1
  3 GrouseShooting in the Highlands                  29
  4 Oxen Ploughing                                   58
  5 A Garden Scene                                   67
  6 The Solitary House                              139
  7 Cattle in the Shade                             164
  8 Sir Roger de Coverley and the Gipsies           165
  9 Ladies personating Gipsies                      195
 10 Daleswomen going to a Shout                     221
 11 Old Dalesman and Traveller                      248
 12 Figures on a Screen in Annesley Hall            286
 13 The Otter Hunt by Bewick                        302
 14 Classical
```

In the picture below we have used re.sub() function to remove the numbers from the book.

```
In [7]: T=re.sub("[0-9]+","",T)
        print(T[:999])
```

```
The


                                                    Page
  Vignette Summerhouse near Claremont               Title
  Old English Hall
  GrouseShooting in the Highlands
  Oxen Ploughing
  A Garden Scene
  The Solitary House
  Cattle in the Shade
  Sir Roger de Coverley and the Gipsies
  Ladies personating Gipsies
 Daleswomen going to a Shout
 Old Dalesman and Traveller
 Figures on a Screen in Annesley Hall
 The Otter Hunt by Bewick
 Classical Rural Scenes
```

4. **lower() –** used to convert a string to lower case.

   Purpose – to convert string to lower case ease of computation.

In the picture below we have used re.sub() function to convert string to lower case.

```
In [8]: T = T.lower()
        print(T[:999])

        the


                                                                  page
           vignette summerhouse near claremont                 title
           old english hall
           grouseshooting in the highlands
           oxen ploughing
           a garden scene
           the solitary house
           cattle in the shade
           sir roger de coverley and the gipsies
           ladies personating gipsies
          daleswomen going to a shout
          old dalesman and traveller
          figures on a screen in annesley hall
          the otter hunt by bewick
          classical rural scenes
```

5. **split() -** is used to break the string specified separator .

   Purpose - Here we are using it to break string into words.

6. **join() -** returns a string concatenated with the elements of iterable.

   Purpose - Here we use it to make braked words into a single string but separated with space.

```
In [9]: T = " ".join(T.split())
        wordcloudT=T
        print(T[:999])

        the page vignette summerhouse near claremont title old english hall grouseshooting in the highlands oxen ploughing a garden sce
        ne the solitary house cattle in the shade sir roger de coverley and the gipsies ladies personating gipsies daleswomen going to
        a shout old dalesman and traveller figures on a screen in annesley hall the otter hunt by bewick classical rural scenes scene i
        n a town street by bewick wild horses in new forest purkiss's cottage new forest charcoalburners' hut wild english cattle in ch
        illingham park woman driving geese procession of village maidens at whitsuntide morgan lewis shewing the last haunt of the fair
        ies the village inn a sea scene a donkey race birdcatching tickling trout i i page preeminence of england as a place of country
        residence its political and moral position the conveniences conferred by the perfection of the arts on social life its literatu
        re spirit of freedom religious feeling and philanthropic institutions the delightfulness of its country resi
```

7. **word_tokenize(String) –** is used to extract token from string of characters.

   Purpose - NOW WE HAVE TO FORMS TOKENS TO PERFORM FUTHER STEPS, Here we have used it to tokenize our previously formed string.

```
In [10]: from nltk.tokenize import word_tokenize
         T = word_tokenize(T)
         print(T[:999])
```

```
['\ufeffthe', 'page', 'vignette', 'summerhouse', 'near', 'claremont', 'title', 'old', 'english', 'hall', 'grouseshooting',
'in', 'the', 'highlands', 'oxen', 'ploughing', 'a', 'garden', 'scene', 'the', 'solitary', 'house', 'cattle', 'in', 'the', 's
hade', 'sir', 'roger', 'de', 'coverley', 'and', 'the', 'gipsies', 'ladies', 'personating', 'gipsies', 'daleswomen', 'going',
'to', 'a', 'shout', 'old', 'dalesman', 'and', 'traveller', 'figures', 'on', 'a', 'screen', 'in', 'annesley', 'hall', 'the',
'otter', 'hunt', 'by', 'bewick', 'classical', 'rural', 'scenes', 'scene', 'in', 'a', 'town', 'street', 'by', 'bewick', 'wil
d', 'horses', 'in', 'new', 'forest', 'purkiss', '’', 's', 'cottage', 'new', 'forest', 'charcoalburners', '’', 'hut', 'wild',
'english', 'cattle', 'in', 'chillingham', 'park', 'woman', 'driving', 'geese', 'procession', 'of', 'village', 'maidens', 'a
t', 'whitsuntide', 'morgan', 'lewis', 'shewing', 'the', 'last', 'haunt', 'of', 'the', 'fairies', 'the', 'village', 'inn',
'a', 'sea', 'scene', 'a', 'donkey', 'race', 'birdcatching', 'tickling', 'trout', 'i', 'i', 'page', 'preeminence', 'of', 'eng
land', 'as', 'a', 'place', 'of', 'country', 'residence', 'its', 'political', 'and', 'moral', 'position', 'the', 'convenience
s', 'conferred', 'by', 'the', 'perfection', 'of', 'the', 'arts', 'on', 'social', 'life', 'its', 'literature', 'spirit', 'o
f', 'freedom', 'religious', 'feeling', 'and', 'philanthropic', 'institutions', 'the', 'delightfulness', 'of', 'its', 'countr
y', 'residences', 'with', 'its', 'parks', 'lawns', 'woods', 'gardens', 'etc', 'the', 'variety', 'of', 'scenery', 'in', 'a',
'small', 'compass', 'advantages', 'of', 'its', 'climate', 'notwithstanding', 'all', 'just', 'cause', 'of', 'complaint', 'it
s', 'soil', 'sanctified', 'by', 'noble', 'deeds', 'and', 'intellectual', 'renown', 'real', 'superiority', 'of', 'england',
'as', 'a', 'place', 'of', 'residence', 'shewn', 'by', 'its', 'effects', 'on', 'foreigners', 'willis', '’', 's', 'descriptio
n', 'of', 'its', 'effect', 'on', 'him', 'enviable', 'position', 'of', 'the', 'english', 'country', 'gentleman', 'as', 'regar
ds', 'all', 'the', 'pleasures', 'and', 'advantages', 'of', 'life', 'every', 'art', 'and', 'energy', 'exerted', 'in', 'his',
'favour', 'by', 'them', 'his', 'house', 'surrounded', 'with', 'delights', 'the', 'news', 'and', 'the', 'luxuries', 'of', 'th
e', 'world', 'brought', 'to', 'his', 'table', 'books', 'music', 'paintings', 'at', 'his', 'command', 'farming', 'gardening',
'planting', 'fieldsports', 'all', 'within', 'his', 'grasp', 'scenes', 'which', 'offer', 'themselves', 'to', 'extend', 'his'
```

Till here we have performed all the necessary preprocessing steps so now we can go ahead and do the required task assign in the project. These steps are same for both the books so we have showed you only one of them.

PART 1

(REFRENCE-->http://www.umiacs.umd.edu/~resnik/temp/pdf/cl.pdf)

Problem Statement – Find the nouns and verbs in both the novels. Get the categories that these words fall under in the WordNet.

We needed to find the noun and verb tagged words to do that we required tokenized list which we have already have from our preprocessing steps.

To find the noun and verb we have done following –

- First we have tagged the tokenized list,
- Then we store the noun and verb both in different arrays,
- To do that we have compared every tagged data with noun and verb symbols respectively.
- Now finally categories obtained array according to the WordNet.

```
In [10]: Noun=[]
         Verb=[]
         PosTags = nltk.pos_tag(T)

         for i in range(len(PosTags)):

             if 'NN' in PosTags[i][1]:
                 Noun.append(PosTags[i][0])
             if 'VB' in PosTags[i][1]:
                 Verb.append(PosTags[i][0])
         print(Noun[:50])
         print(Verb[:50])
```

Here, we have every element with NN for noun because it will include every other form like NNS, NNP because it also contains NN as substring in it.

Similarly with verb we have compared VB but it will include every other form like VBZ, VBN, VBP etc. in the list that will be made from the array.

The Functions used –

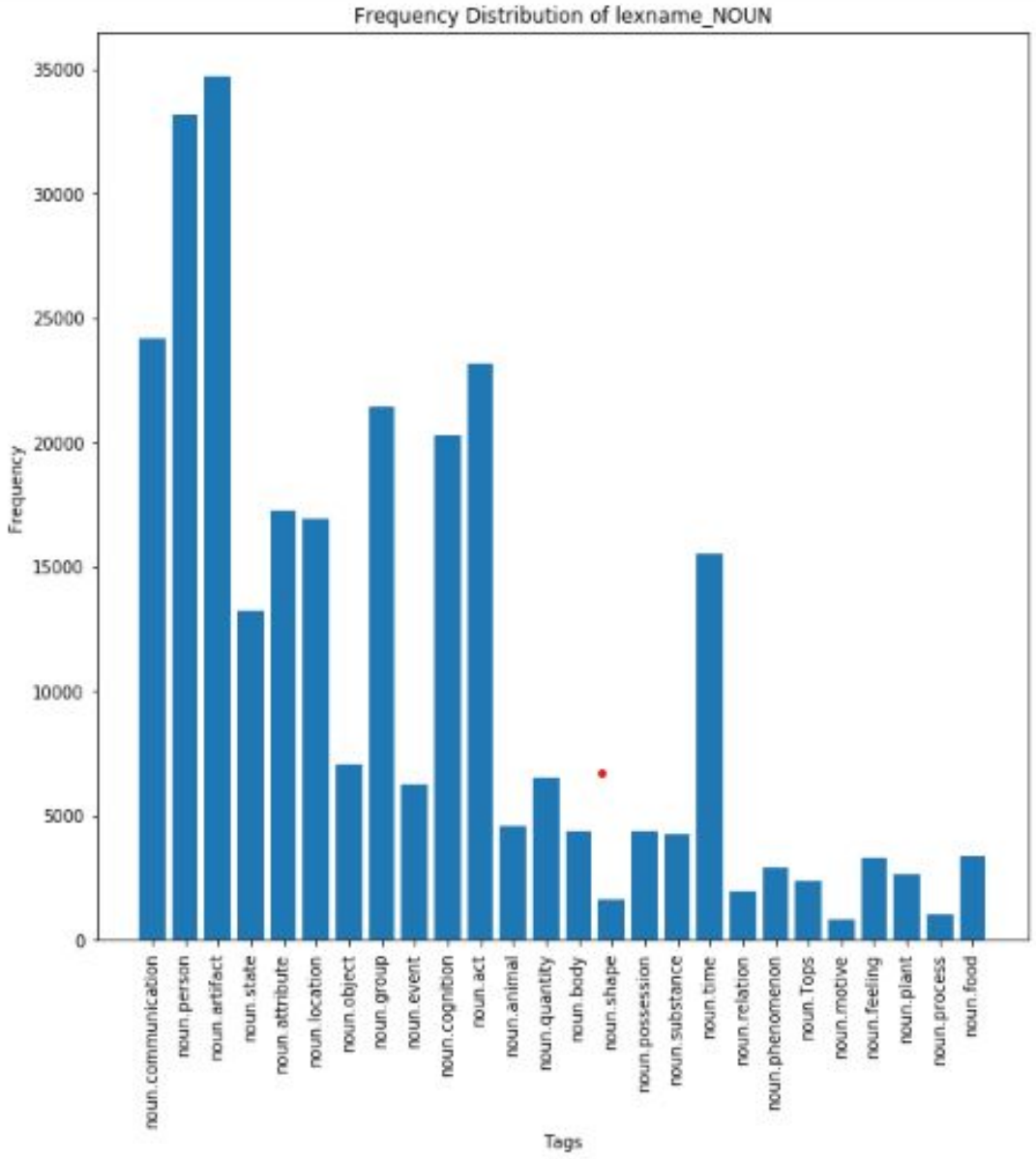
- **Synsets⟶** Synset is a special kind of a simple interface that is present in NLTK to look up words in WordNet. Synset instances are the groupings of synonymous words that express the same concept.
- **Lexname ⟶** The file lexnames lists the mapping between file names and numbers, and can be used by programs or end users to correlate the two.

Noun –

```
In [11]: lex_n=[]
         lexi_n=[]
         tagi_n={}
         ######first found the synsets if each word and merged them in one list(lex) simultaneously#######
         for i in range(len(Noun)):
             lex_n+= wn.synsets(Noun[i],'n')
         ######found the lexname of each word corresponding there synsets and stored them in new list--> lexi ######
         for i in range(len(lex_n)):
             lexi_n.append(lex_n[i].lexname())
         ######found the frequency of the lexnames that appeared in the text having Noun As POS tag #########
         for i in range(len(lexi_n)):
             if lexi_n[i] not in tagi_n:
                 tagi_n[lexi_n[i]] = 1

             else:
                 tagi_n[lexi_n[i]] += 1
         keys = list(tagi_n.keys())
         values = list(tagi_n.values())

         plt.figure(figsize=(10,10))
         plt.bar(keys,values)
         plt.xlabel("Tags")
         plt.xticks(rotation = 90)
         plt.ylabel("Frequency")
         plt.title("Frequency Distribution of lexname_NOUN")
         plt.show()
```

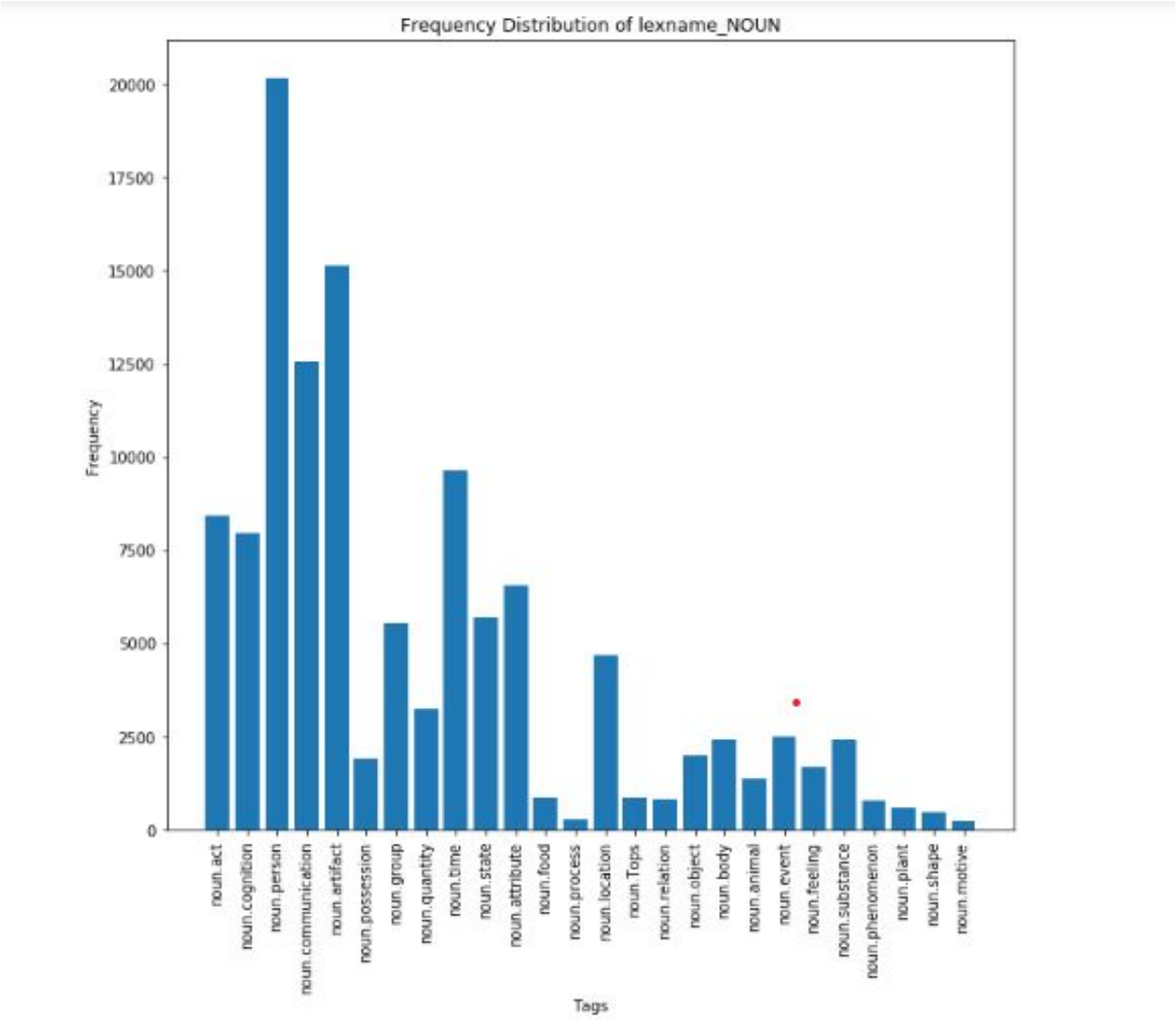
Code explained – every step is commented explaining the reason behind every step

Noun plot for BOOK – The rural Life of England



The above plot shows the noun categorized words and there frequencies according to the WordNet

Noun plot for BOOK – The book of the thousand nights and a night (volume 3)



The above plot shows the noun categorized words and there frequencies according to the WordNet

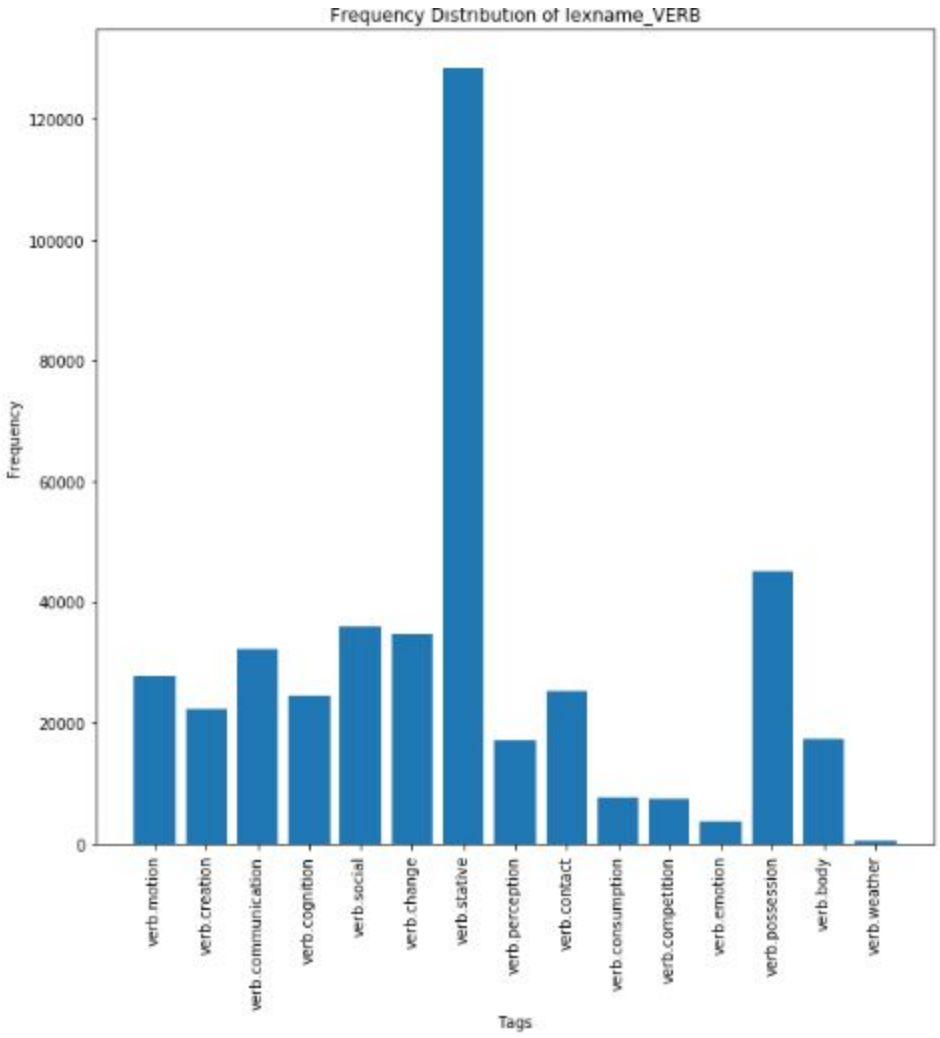
Generally there are 25 categories in noun and the 26th one is 'Tops' in both the books. few generic nouns used were stored in 'Tops' file to relate the several tropical files.

VERB-

In [12]:
```
lex_v=[]
lexi_v=[]
tagi_v={}
######first found the synsets of each word and merged them in one list(lex) simultaneously#######
for i in range(len(Verb)):
    lex_v+= wn.synsets(Verb[i],'v')
######found the lexname of each word corresponding there synsets and stored them in new list--> lexi ######
for i in range(len(lex_v)):
    lexi_v.append(lex_v[i].lexname())
######found the frequency of the lexnames that appeared in the text having Verb As POS tag #########
for i in range(len(lexi_v)):
    if lexi_v[i] not in tagi_v:
        tagi_v[lexi_v[i]] = 1

    else:
        tagi_v[lexi_v[i]] += 1
keys = list(tagi_v.keys())
values = list(tagi_v.values())

plt.figure(figsize=(10,10))
plt.bar(keys,values)
plt.xlabel("Tags")
plt.xticks(rotation = 90)
plt.ylabel("Frequency")
plt.title("Frequency Distribution of lexname_VERB")
plt.show()
```

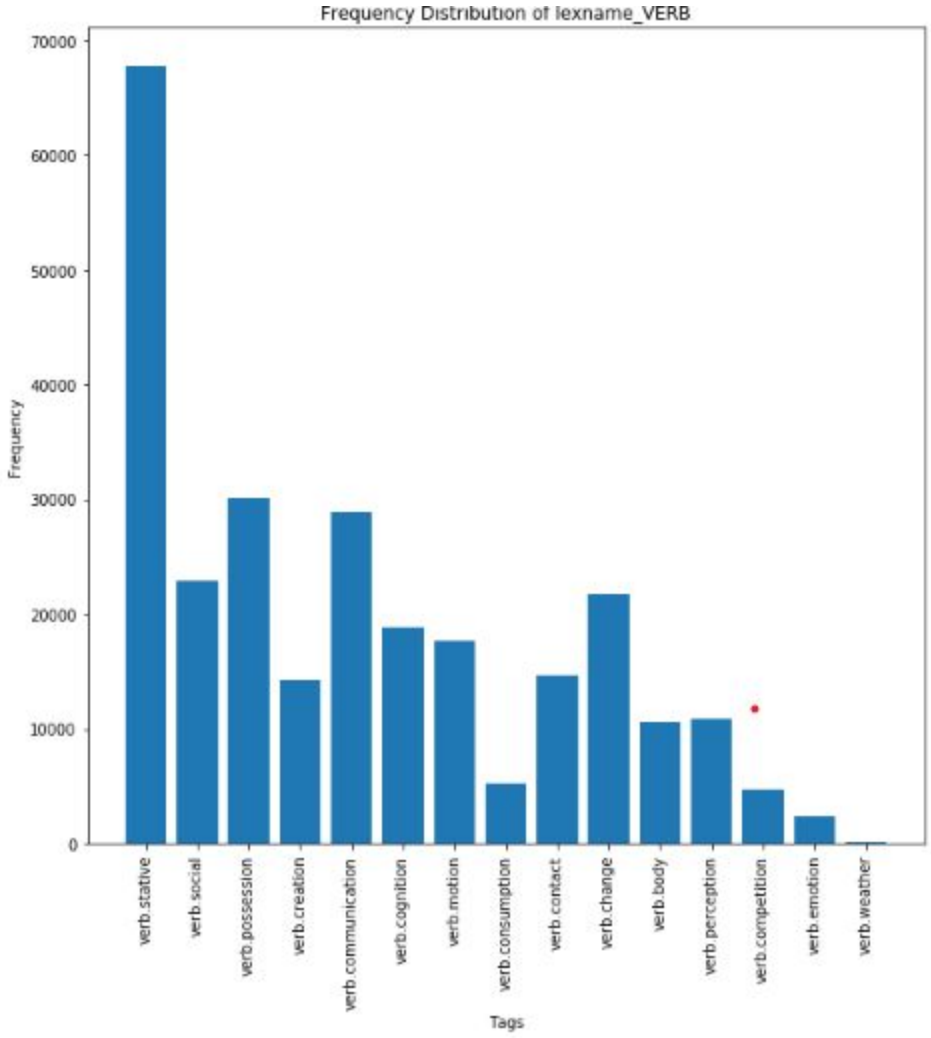
We can use the same code we have used for noun to get plot for verb.

Verb plot for BOOK – The rural Life of England



The above plot shows the verb categorized words and there frequencies according to the WordNet.

Verb plot for BOOK – The book of the thousand nights and a night (volume 3)



The above plot shows the verb categorized words and there frequencies according to the WordNet.

PART-2

Refrence for this section
-->https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da

The Problem statement was divided into two parts:

i) Recognise all the entity.

```
In [32]: import spacy
         from spacy import displacy
         from collections import Counter
         import en_core_web_sm
         nlp = spacy.load('en_core_web_sm')
         nlp.max_length=1500000
         doc1 = nlp(book1)
```

```
doc2 = nlp(book2)
```

We first imported Spacy library. This library provides one-stop-shop for many purposes like POS tagging,Entity Recognition,Tokenization,etc. Here we will use it for entity recognition purpose.After importing Spacy we need to load Spacy's pipeline which is stored in a variable '**nlp**' and the model that is loaded is **en_core_web_sm(**an english language based model**).** We invoked **nlp** on the books we are using- "The Rural England" and "The Thousand Nights and Nights" to create a doc1 and doc2 respectively, the doc1 and doc2 are now vessels for NLP tasks on the book itself.

```
print(len(doc1.ents))
```

```
9681
```

BOOK1

```
print(len(doc2.ents))
```

```
6172
```

BOOK2

The total number of entities types present(not distinct) in the respective books.

```
print([(X.text) for X in doc1.ents][:20])
```

```
['William Howitt', 'the United States', 'the Project Gutenberg License', 'eBook', 'the United States', 'The Rural Life of Engla
nd\n\n', 'William Howitt', 'Thomas Bewick', 'S. Williams', 'October 18, 2019', '60485', 'English', 'Chris Curnow', 'Harry Lam
é', 'The Internet Archive', '’s Notes', 'Text', 'Fraktur', '~tildes~. Small', 'Transcriber’s Notes']
```

BOOK1

```
print([(X.text) for X in doc2.ents][:20])
```

```
['Richard F. Burton', 'the United States', 'the Project Gutenberg License', 'eBook', 'the United States', 'The Book of the Thou
sand Nights and a Night -- Volume 3', 'Richard F. Burton', 'December 11, 2019', '60889', 'English', 'Richard Tonsing', 'Richard
Hulse', 'The Internet Archive', 'TO THE PURE ALL THINGS ARE PURE', 'Puris omnia pura', 'Arab Proverb', 'intese mai sanamente pa
role', '_Decameron_”–_conclusion_', 'Erubuit', 'Lucretia']
```

BOOK2

Here we have entity present in the books(book1 and book2) of the first 20 lines.

ii) Recognise all entity types.

```
labels = [x.label_ for x in doc1.ents]
Counter(labels)
```

```
Counter({'PERSON': 1901,
         'GPE': 1480,
         'ORG': 1544,
         'LOC': 194,
         'DATE': 1288,
         'MONEY': 17,
         'NORP': 433,
         'WORK_OF_ART': 171,
         'ORDINAL': 191,
         'CARDINAL': 1644,
         'FAC': 98,
         'LANGUAGE': 112,
         'PRODUCT': 115,
         'QUANTITY': 156,
         'LAW': 15,
         'TIME': 296,
         'EVENT': 23,
         'PERCENT': 3})
```

BOOK1

```
labels1 = [x.label_ for x in doc2.ents]
Counter(labels1)
```

```
Counter({'PERSON': 1701,
         'GPE': 731,
         'ORG': 846,
         'LOC': 131,
         'WORK_OF_ART': 281,
         'DATE': 402,
         'MONEY': 4,
         'NORP': 496,
         'TIME': 236,
         'CARDINAL': 989,
         'LANGUAGE': 24,
         'ORDINAL': 160,
         'PRODUCT': 78,
         'FAC': 57,
         'QUANTITY': 14,
         'EVENT': 15,
         'LAW': 6,
         'PERCENT': 1})
```

BOOK2

Here we have all entity types present in the books respectively including its frequency.

ADDITIONAL EXERCISES

Extract the relationship between entities.

X.text⟶ it gives me the entity.

X.label_⟶ it gives me the entity type associated.

```
print([(X.text, X.label_) for X in doc1.ents][:20])
```

```
[('William Howitt', 'PERSON'), ('the United States', 'GPE'), ('the Project Gutenberg License', 'ORG'), ('eBook', 'LOC'), ('the
United States', 'GPE'), ('The Rural Life of England\n\n', 'ORG'), ('William Howitt', 'PERSON'), ('Thomas Bewick', 'PERSON'),
('S. Williams', 'PERSON'), ('October 18, 2019', 'DATE'), ('60485', 'MONEY'), ('English', 'NORP'), ('Chris Curnow', 'PERSON'),
('Harry Lamé', 'PERSON'), ('The Internet Archive', 'ORG'), ('’s Notes', 'PERSON'), ('Text', 'ORG'), ('Fraktur', 'ORG'), ('~tild
es~. Small', 'ORG'), ('Transcriber’s Notes', 'WORK_OF_ART')]
```

BOOK1

```
print([(X.text, X.label_) for X in doc2.ents][:20])
```

```
[('Richard F. Burton', 'PERSON'), ('the United States', 'GPE'), ('the Project Gutenberg License', 'ORG'), ('eBook', 'LOC'), ('t
he United States', 'GPE'), ('The Book of the Thousand Nights and a Night -- Volume 3', 'WORK_OF_ART'), ('Richard F. Burton', 'P
ERSON'), ('December 11, 2019', 'DATE'), ('60889', 'MONEY'), ('English', 'NORP'), ('Richard Tonsing', 'PERSON'), ('Richard Huls
e', 'PERSON'), ('The Internet Archive', 'ORG'), ('TO THE PURE ALL THINGS ARE PURE', 'WORK_OF_ART'), ('Puris omnia pura', 'PERSO
N'), ('Arab Proverb', 'PERSON'), ('intese mai sanamente parole', 'GPE'), ('_Decameron_”–_conclusion_', 'WORK_OF_ART'), ('Erubui
t', 'WORK_OF_ART'), ('Lucretia', 'ORG')]
```

BOOK2

In the above pictures we were working on entity level, in the following pictures, we are demonstrating token-level entity annotation using the **BILUO** (B-Beginning,I-Inside,L-last,O-outside,U-unit) tagging scheme to describe the entity boundaries.

```
print([(X, X.ent_iob_, X.ent_type_) for X in doc1][5000:5050])
```

```
[(by, 'O', ''), (many, 'O', ''), (of, 'O', ''), (
  , 'O', ''), (our, 'O', ''), (Family, 'B', 'ORG'), (Pictures, 'I', 'ORG'), (--, 'O', ''), (Treasures, 'O', ''), (of, 'O', ''),
(Ancient, 'O', ''), (Art, 'O', ''), (collected, 'O', ''), (in, 'O', ''), (our, 'O', ''), (
  , 'O', ''), (Noble, 'B', 'PERSON'), (Houses, 'I', 'PERSON'), (--, 'O', ''), (Horace, 'O', ''), (Walpole, 'O', ''), (’s, 'O',
''), (Wish, 'O', ''), (,, 'O', ''), (that, 'O', ''), (all, 'O', ''), (our, 'O', ''), (Noble, 'B', 'PRODUCT'), (Mansions, 'I',
'PRODUCT'), (
  , 'O', ''), (were, 'O', ''), (congregated, 'O', ''), (in, 'O', ''), (London, 'B', 'GPE'), (--, 'O', ''), (beneficial, 'O',
''), (Influence, 'O', ''), (of, 'O', ''), (the, 'O', ''), (Country, 'O', ''), (
  , 'O', ''), (Residence, 'O', ''), (of, 'O', ''), (the, 'O', ''), (Aristocracy, 'O', ''), (--, 'O', ''), (Feelings, 'O', ''),
(of, 'O', ''), (Horace, 'O', ''), (Walpole, 'O', '')]
```

BOOK1

```
print([(X, X.ent_iob_, X.ent_type_) for X in doc1][5000:5050])
```

```
[(permanence, 'O', ''), (in, 'O', ''), (prosperity.—And, 'B', 'ORG'), (Shahrazad, 'I', 'ORG'), (was, 'O', ''), (surprised,
'O', ''), (by, 'O', ''), (the, 'O', ''), (
, 'O', ''), (dawn, 'O', ''), (of, 'O', ''), (day, 'O', ''), (and, 'O', ''), (ceased, 'O', ''), (to, 'O', ''), (say, 'O', ''),
(her, 'O', ''), (permitted, 'O', ''), (say, 'O', ''), (., 'O', ''), (


       , 'O', ''), (Now, 'O', ''), (when, 'O', ''), (it, 'O', ''), (was, 'O', ''), (the, 'O', ''), (Four, 'B', 'CARDINAL'), (Hu
ndred, 'I', 'CARDINAL'), (and, 'I', 'CARDINAL'), (Ninety, 'I', 'CARDINAL'), (-, 'I', 'CARDINAL'), (eighth, 'O', ''), (Night,
'O', ''), (,, 'O', ''), (

, 'O', ''), (Quoth, 'O', ''), (Dunyazad, 'O', ''), (,, 'O', ''), (“, 'O', ''), (O, 'O', ''), (sister, 'O', ''), (mine, 'O',
''), (,, 'O', ''), (an, 'O', ''), (thou, 'O', ''), (be, 'O', ''), (other, 'O', ''), (than, 'O', ''), (sleepy, 'O', ''), (,,
'O', '')]
```

BOOK2

display.render-->Render a dependency parse tree or named entity visualization.We have run (displacy.render) to generate the markup.

We have two style:

1. **'ent'**--> The entity visualizer,highlights named entities and their labels in a text. we use both of them.

Extracted the relationship between the entities....

```
displacy.render(nlp(book1[9000:10000]), jupyter=True, style='ent')
```

221 CARDINAL 11 CARDINAL . Old Dalesman PERSON and Traveller 248 12 CARDINAL . Figures on a Screen ORG in Annesley Hall FAC 286 13 CARDINAL . The Otter Hunt WORK_OF_ART , by Bewick ORG 302 CARDINAL 14 CARDINAL . Classical Rural Scenes 305 15 CARDINAL . Scene in a Town Street FAC , by Bewick 324 ORG 16 CARDINAL . Wild Horses in New Forest GPE 366 CARDINAL 17 CARDINAL . Purkiss's Cottage, New Forest ORG 376 CARDINAL 18 CARDINAL . Charcoal-burners' Hut 379 CARDINAL 19 CARDINAL . Wild English Cattle in Chillingham Park ORG 395 CARDINAL 20 CARDINAL . Woman driving Geese ORG 431 CARDINAL 21 CARDINAL . Procession of Village Maidens at Whitsuntide 444 CARDINAL 22 CARDINAL . Morgan Lewis PERSON shewing the last haunt of the Fairies 479 CARDINAL 23 CARDINAL . The Village Inn 480 CARDINAL 24 CARDINAL . A Sea Scene 502

Hence we can see that the entities are marked.

BOOK1

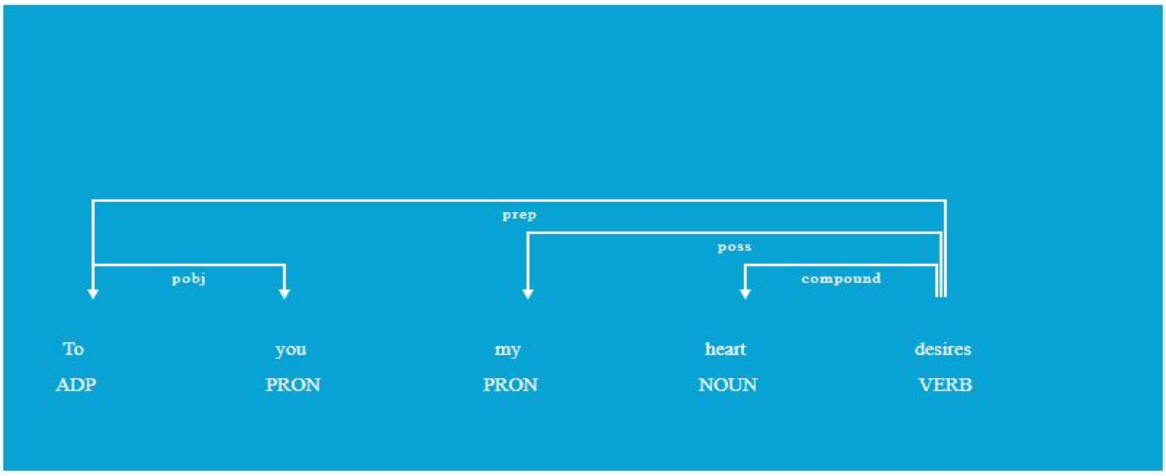
```
displacy.render(nlp(book2[8000:10000]), jupyter=True, style='ent')
```

rage à part of200–300 copies entitled _Histoire ORG d' ' Alá al-Dîn ou La Lampe Merveilleuse PERSON , Texte Arabe PERSON , publié par H. Zotenberg PERSON ; Paris GPE , Imprimerie Nationale ORG , 1888 DATE _; including a most important contribution:—_Sur quelquesManuscrits des Mille PERSON et une Nuits et la traduction de Galland_.[1]The learned and genial author has favoured me with proof sheets of hislabours: it would be unfair to disclose the discoveries, such as the Manuscript Journals ORG in the Bibliothèque Nationale (Nos. ORG 15277 to 15280 DATE),which the illustrious Galland PERSON kept regularly till the end of his life,and his conversations with " M. Hanna PERSON , Maronite d'Halep ORG ," alias Jean Dipi PERSON (Dippy, a corruption of Diab ORG): suffice it to say that they cast a clearand wholly original light upon the provenance of eight CARDINAL of the Gallandian NORP histories. I can, however, promise to all " Aladdinists WORK_OF_ART " a rich harvestof facts which wholly displace those hitherto assumed to be factual. Butfor the satisfaction of my readers I am compelled to quote the colophonof M. Zotenberg PERSON 's great "find" (vol. ii.), as it bears upon a highlyimportant question. " And the finishing thereof WORK_OF_ART was during the first decade DATE of Jamádi PERSON the Second ORDINAL , of the one thousand and one hundred CARDINAL and fifteenth year of the Hegirah (= A.D. 1703) by the transcription of the neediest of His slaves unto Almighty Allah, Ahmad bin Mohammed al-Tarádí PERSON , in Baghdad City GPE : he was a Sháfi'í of school, and a Mosuli GPE by birth, and a Baghdadi by residence, and he wrote it for his own use, and upon it he imprinted his signet. So Allah save our lord Mohammed PERSON and his Kin and Companions WORK_OF_ART and assain them! Kabíkaj PERSON ."[2]Now as this date corresponds with A.D. 1703 PERSON , whereas Galland ORG did notbegin publishing until 1704–1705 CARDINAL , the original MS. of Ahmad al-Tarádí PERSON could not have been translated or adapted from the French NORP ; and althoughthe transcription by Mikhail Sabbagh PERSON , writing in 1805–10 GPE ,

BOOK2

2. ‘dep’--> The dependency visualizer,shows part-of-speech tags and syntactic dependencies.

```
options = {"compact": True, "bg": "#09a3d5",
           "color": "white", "font": "Source Sans Pro"}
displacy.render(nlp(book1[2002:2025]), jupyter=True, style='dep',options=options)
```



BOOK1

In [9]:
```
options = {"compact": True, "bg": "#09a3d5",
           "color": "white", "font": "Source Sans Pro"}
displacy.render(nlp(book2[3017:3055]), jupyter=True, style='dep',options=options)
```