NLP Project Report

Under the Guidance of Dr. Sakthi Balan

Team name: BitGraders

Github Repo Link :
https://github.com/ShivamMundhra/NLP-Project

Members :

Atulya Singh (18ucs166)
Akshay Solanki (18ucs225)
Shivam Mundhra (18ucs012)
Govind Singh Shekhawat (18ucs003)

INDEX

System Requirements:

1. Python 3.x
2. Libraries : NLTK, matplotlib, wordcloud, re, NLTK.corpus, json, ssl, urllib
3. Jupyter notebook
4. OS: Windows / Linux

Books Used:

Book-1 :

Title : The Flying Boys to the Rescue
Author : Edward S. Ellis
Illustrator : Edwin J. Prittie
No. of words : *59,746*
No. of lines : *7,647*
No. of characters (with spaces) : *3,62,698*
Link : http://www.gutenberg.org/files/63365/63365-0.txt

Book-2 :

Title : The Sense of the Past
Author : Henry James
Contributor : Percy Lubbock
No. of words : *84,714*
No. of lines : *8,228*
No. of characters (with spaces) : *5,17,554*
Link : http://www.gutenberg.org/files/63369/63369-0.txt

Project Round-1

Problem statement - 1 :

Import the text, lets call it as data1 and data2.

We have imported Book-1 as data1 and Book-2 as data2. We have used **"*json*"** and **"*urllib*"** libraries to scrap the texts from the given links. Further we decoded it to convert the text to string.

Problem statement - 2 :

Perform simple text pre-processing.

1. Changing the text to lowercase.
2. Removing running sections.
3. Removing special characters.
4. Removing Chapter Headlines.
5. Removing digits.
6. Removing implicit next line characters - '\n'.
7. Expansion of contractions like *"aren't"* to "*are not*", *"we've"* to *"we have"*, etc.

Problem statement - 3 :

Tokenize the texts.

Tokenization is basically dividing the text into smaller units called tokens. They can be either words, characters, or subwords. It helps in understanding the context and interpreting the meaning behind the sequence of tokens. For example "*Book that flight",* will be tokenized into - ["*Book*", "*that*", "*flight*"].

- '*data1*' and '*data2*' are tokenized using the ***nltk.word_tokenize*** function.
- **'*token1*'** contains tokens of '*data1*' i.e. Book-1.
- '***token2*'** contains tokens of '*data2*' i.e. Book-2.

Inference:

So now the imported texts **'*data1*'** and **'*data2*'** will be tokenized as explained above using ***nltk.word_tokenize*** function. Tokens created are now stored in '*token1*' for '*data1*' and '*token2*' for '*data2*'.

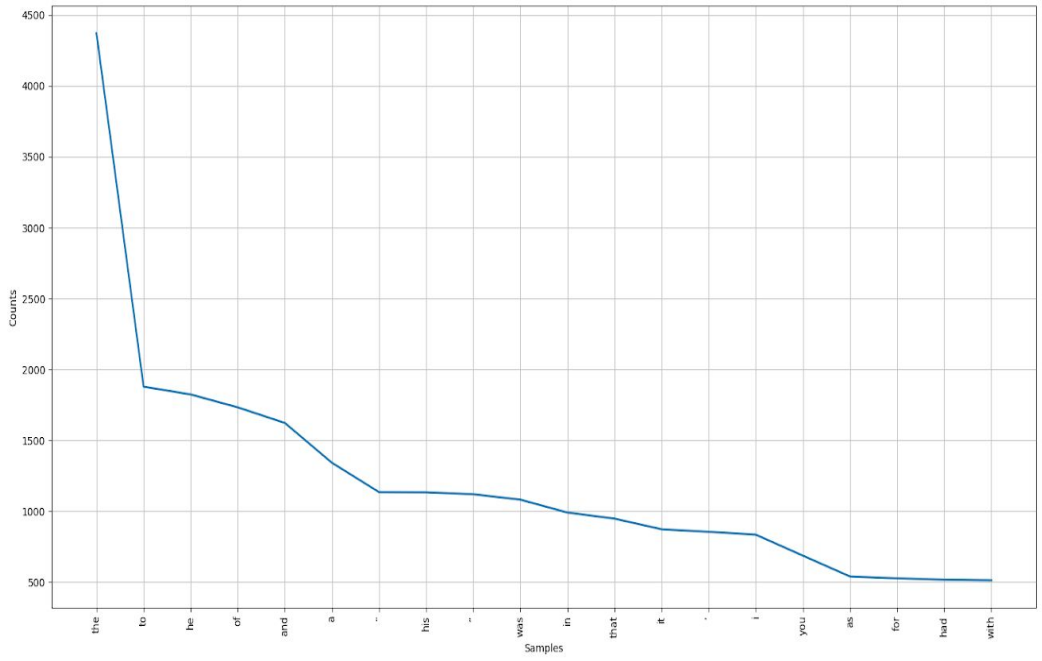
Problem statement - 4 :

Analyze the frequency distribution of tokens separately.

Formed the Frequency Distribution using FreqDist function of NLTK library and then plotted it using *matplotlib*.

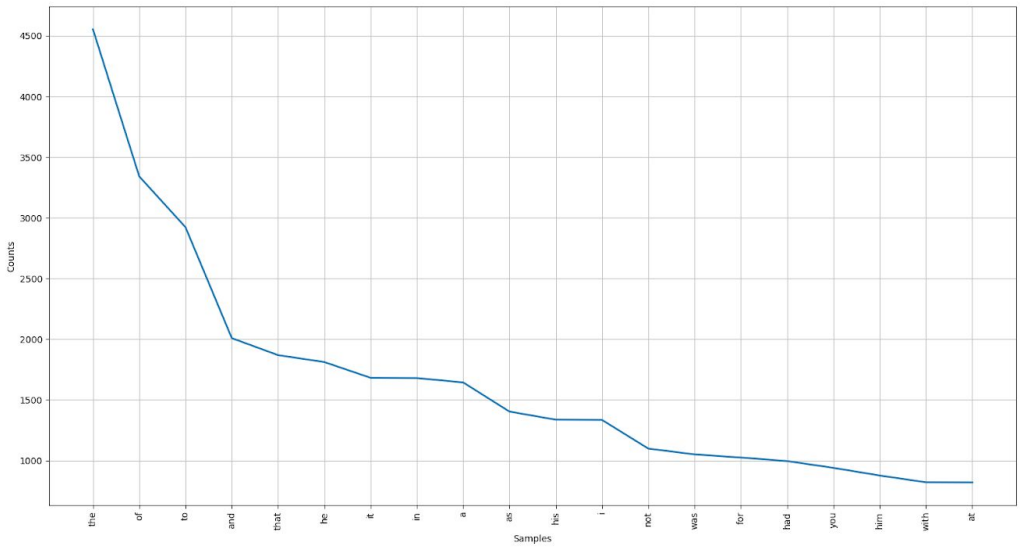
Token1:

```
Extracting  http://www.gutenberg.org/files/63369/63369-0.txt
<FreqDist with 6472 samples and 68497 outcomes>
[('the', 4372), ('to', 1879), ('he', 1824), ('of', 1733), ('and', 1623), ('a', 1342), ('
“', 1135), ('his', 1134), ('”', 1120), ('was', 1082), ('in', 991), ('that', 948), ('it',
 873), ('’', 855), ('i', 835), ('you', 687), ('as', 540), ('for', 527), ('had', 517), ('
with', 513)]
```



Token2:

```
<FreqDist with 8176 samples and 93009 outcomes>
[('the', 4554), ('of', 3342), ('to', 2923), ('and', 2009), ('that', 1869), ('he', 1812),
 ('it', 1682), ('in', 1680), ('a', 1644), ('as', 1404), ('his', 1337), ('i', 1335), ('no
t', 1099), ('was', 1051), ('for', 1025), ('had', 996), ('you', 940), ('him', 877), ('wit
h', 821), ('at', 820)]
```



Problem statement - 5 :

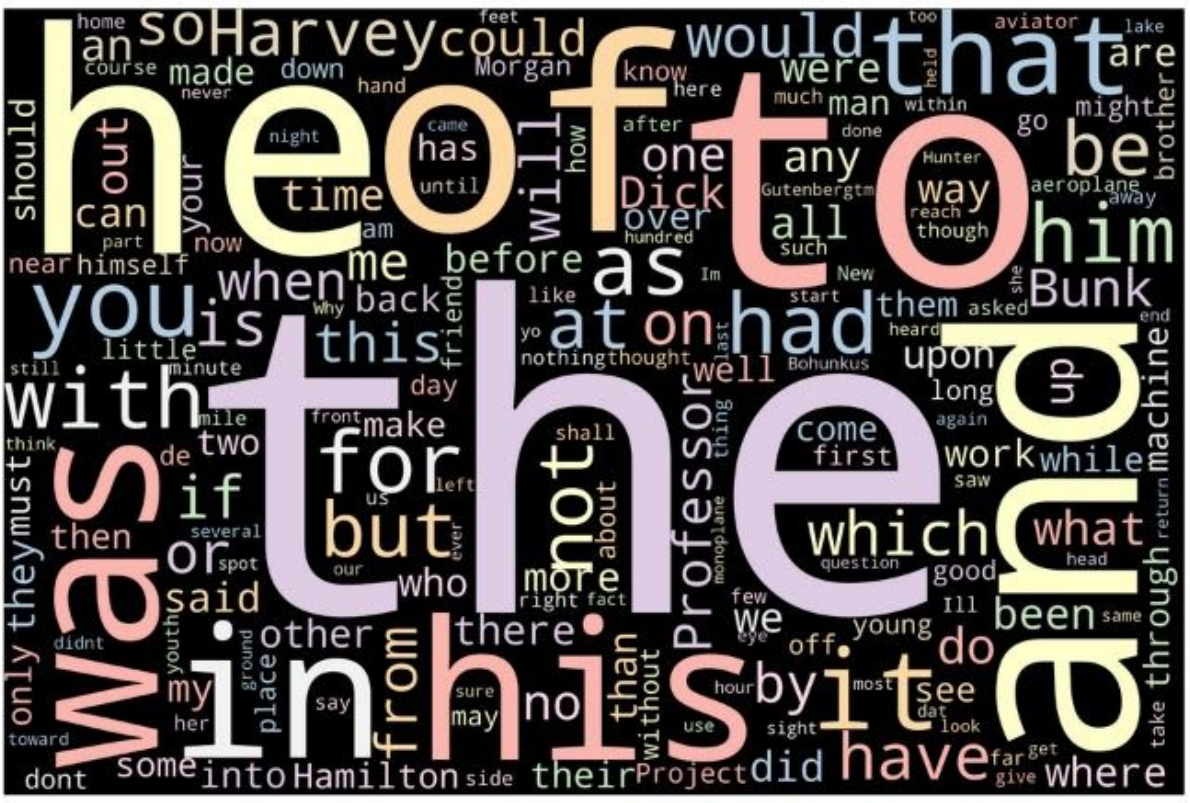
Create a wordcloud before and after removing the stop words.

After analyzing the frequency distribution of the tokens, wordclouds are created using these tokens. A wordcloud is a visual representation of text data wherein each word is pictured with its frequency, i.e. higher the frequency, larger the size of the word.

Stopwords - These are most commonly occurring words because of the grammar rules which hinder the visualization and useful computations.

Before removing stopwords :

Book-1 :



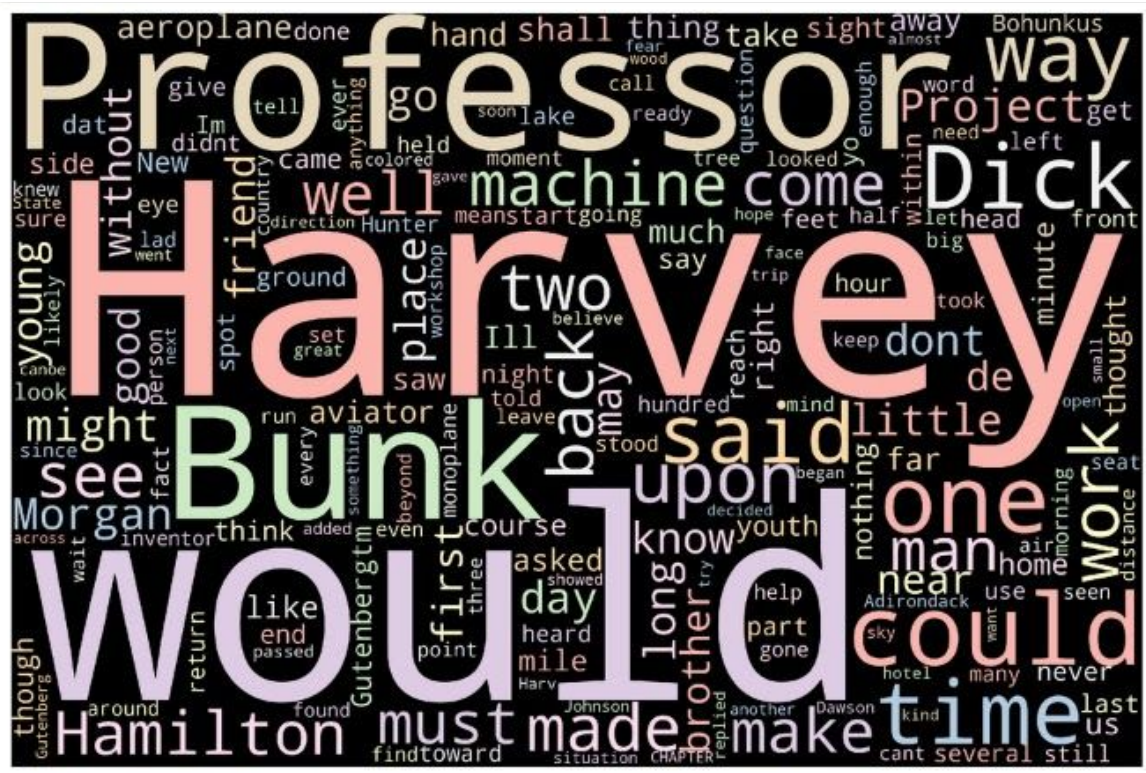
Book-2:

the
to
of
in
that
and
as
his
it
was
with
have
had
her
for
he
you
which
by
be
not
but
him
at
so
is
what
she
this
all
on
an
would
from
Ralph
if
more
were
one
time
how
up
my
we
might
out
own
yet
me
much
than
other
or
now
then
any
see
been
little
before
could
should
make
only
your
do
least
fact
first
such
when
them
they
know
even
there
just
about
himself
their
dont
very
our
no
say
effect
though
shall
want
said
take
something
like
come
sense
hand
young
while
having
moment
after
Im
didnt
too
again
once
way
great
face
back
most
must
however
point
mean
good
some
thus
really
quite
upon
give
man
being
last
put
friend
seemed
thing
anything
part
word
still
where
from
case
instant
whatever
Molly
herself
world
question
enough
less
into
Ambassador
ever
think
almost
house
made
course
has
seem
may
rather
itself
under
are
perhaps
off
feel
felt
these
work
right
can
get
matter
interest
without
here
already
Perry
Sir
us
same
Cantopher
Mr
place
Yes
mother
Midmore
taken
eye
indeed
look
another
somehow
far
two
never
Project
find
will
over
nothing
fine

After removing stopwords :

NLTK library provides a list of the stopwords of the english language. So with the help of this list, we omitted out the stopwords from our tokens.

Book-1 :



Book-2 :

Inference :

As it can be clearly seen in above pictures of wordcloud before removing the stopwords, the larger words are *‘he’*, *‘of’*, *‘the’*, *‘to’*, etc. These are the stopwords which occur very frequently (size of a word in a wordcloud is determined according to its frequency) and these words are not essential for our analysis. So after removing the stopwords, we can ensure that other words which are in context with our books show up in the wordcloud.

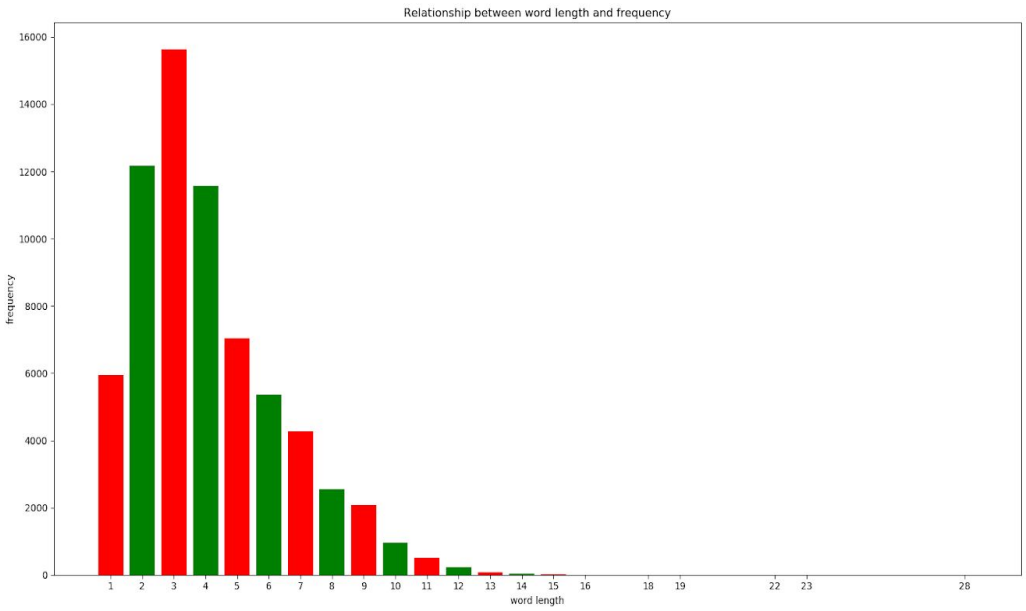
Problem statement - 6 :

Evaluate the relationship between the word length and frequency after removing stopwords.

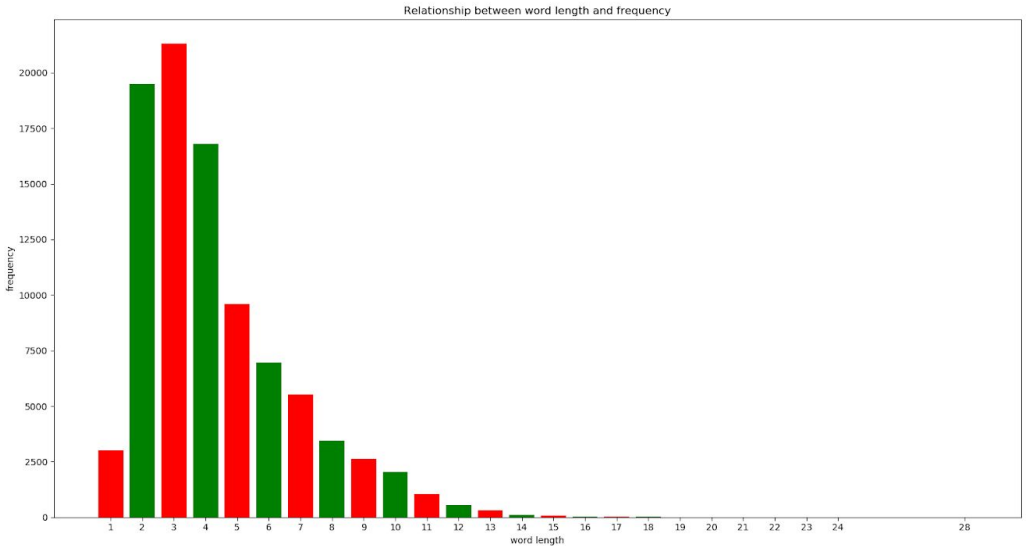
Firstly we created an empty list and stored the number of words for each length and then plotted it using Frequency Distribution and matplotlib.

Before Removing Stopwords:-

Book-1:

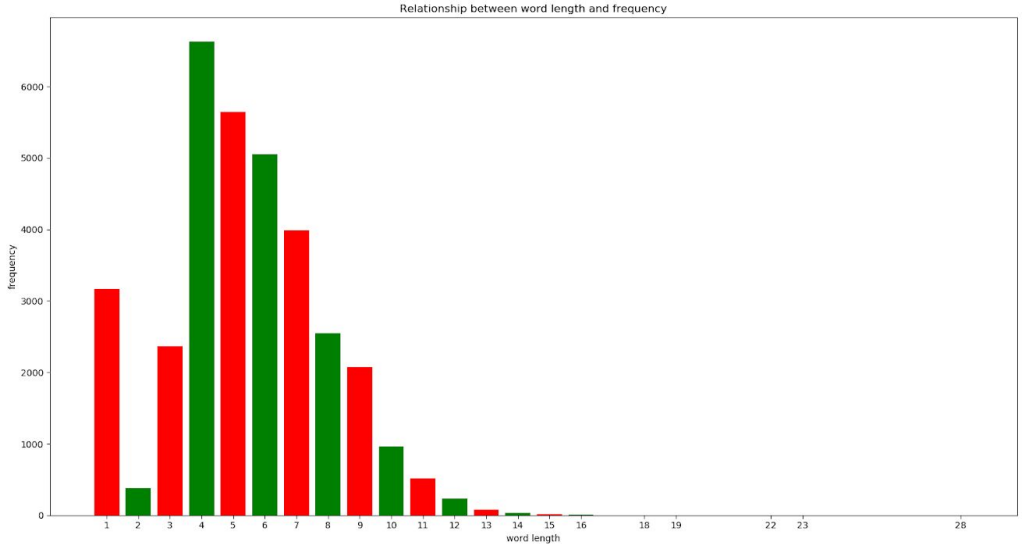


Book-2:

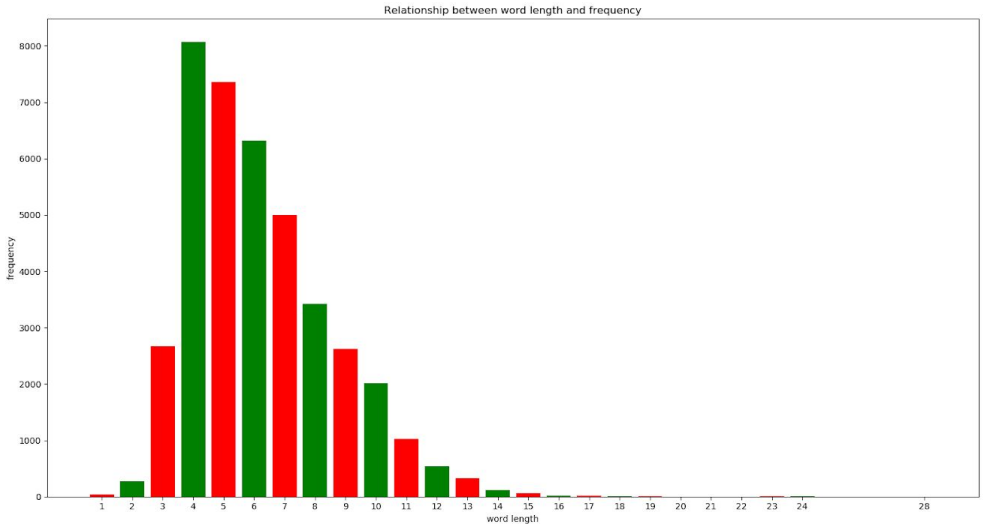


After Removing Stopwords:-

Book-1:



Book-2:



Inference :

Frequency of words of smaller length decreased significantly after removing the stopwords as generally these stopwords are of smaller length, e.g. - *'to', 'the',* etc.

Problem statement - 7 :

Do PoS Tagging using anyone of the four tagset studied in the class and get the distribution of various tags.

PoS stands for parts of speech. PoS tagging means to tag each word with its part of speech like noun, verb, adverb, adjective, etc. Though there are some third party libraries which can be used to do PoS tagging in just one line of code, there is a very fancy algorithm which is doing all the work behind this. Basically it uses *HMM (hidden markov model)* to tag each word. The crux of the algorithm is to find the most probable tagset for the given set of words among all the possible tagsets. It uses the *Viterbi* algorithm which is based on the dynamic programming paradigm to efficiently find the most probable tagset. There are many available tagsets which can be used for this purpose. For this project, we have used the *Treebank* tagset. *Treebank* tagset includes *36* PoS tags like Coordinating Conjunction (CC), Determiner (DT) etc.

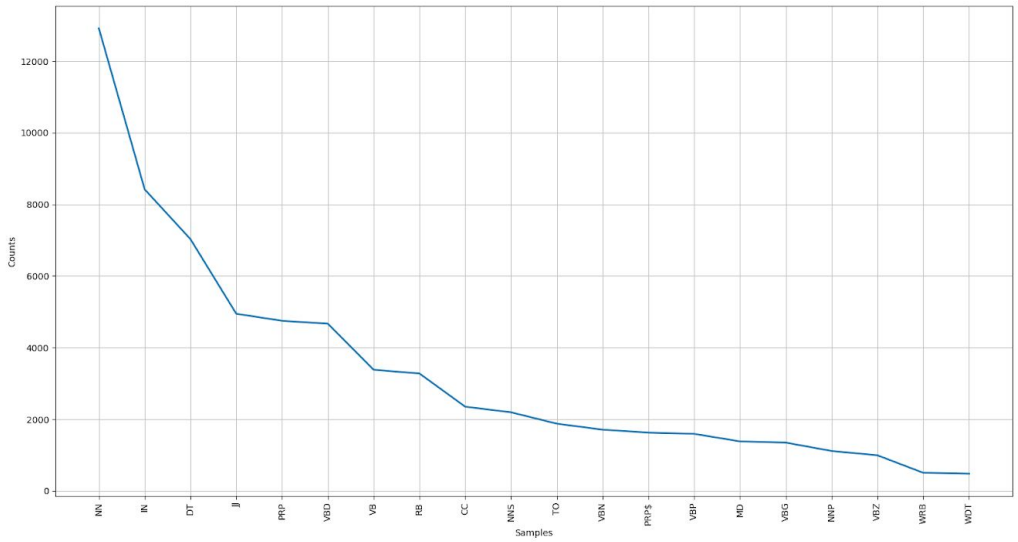
Note :- Below plots are for the 20 most occuring tags.

Book-1 :-

PoS tagging analysis

```
<FreqDist with 34 samples and 68497 outcomes>
[('NN', 12914), ('IN', 8420), ('DT', 7030), ('JJ', 4948), ('PRP', 4751), ('VBD', 4670),
('VB', 3384), ('RB', 3280), ('CC', 2353), ('NNS', 2197), ('TO', 1879), ('VBN', 1712), ('
PRP$', 1630), ('VBP', 1595), ('MD', 1385), ('VBG', 1351), ('NNP', 1117), ('VBZ', 997), (
'WRB', 509), ('WDT', 482)]
```

PoS tagging plot



Book-2 :

PoS tagging analysis

```
<FreqDist with 33 samples and 93009 outcomes>
[('NN', 14374), ('IN', 14298), ('DT', 8660), ('RB', 7456), ('PRP', 7386), ('JJ', 6174),
('VBD', 4709), ('VB', 4441), ('CC', 3055), ('PRP$', 3048), ('TO', 2923), ('VBN', 2740),
('NNS', 2411), ('VBP', 2141), ('MD', 1724), ('VBG', 1567), ('VBZ', 1431), ('WDT', 859),
('WP', 683), ('CD', 484)]
```

PoS tagging plot