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’.

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:

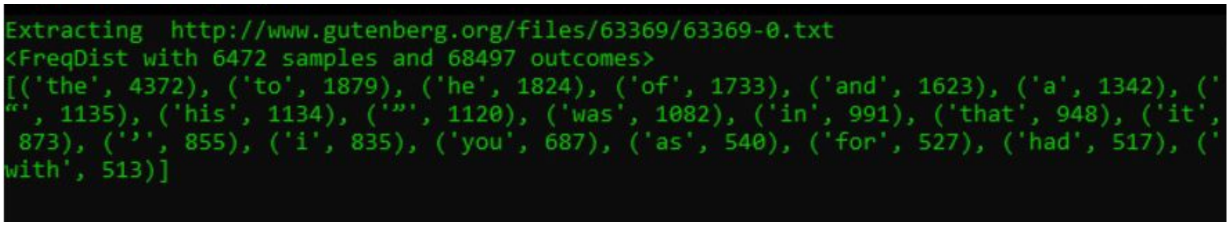
So now the imported texts ***‘data1’*** and ***‘data2’*** will be tokenized as explained above using ***nltk.word_tokenize*** function. Tokens created are now stored in ‘*token1*’ for ‘*data1*’ and ‘*token2*’ for ‘*data2*’.

Problem statement - 4 :

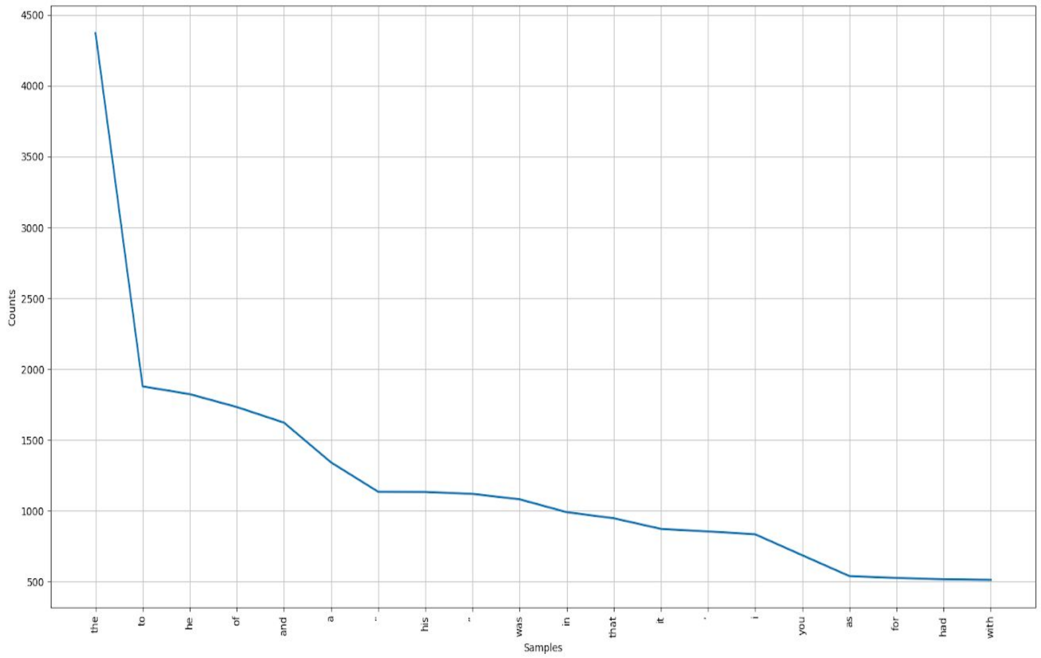
Analyze the frequency distribution of tokens separately.

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

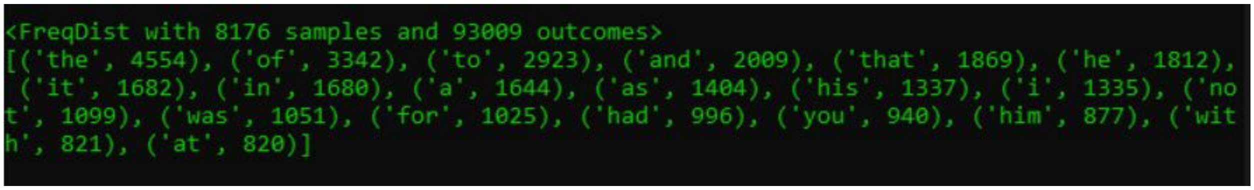
Token1:



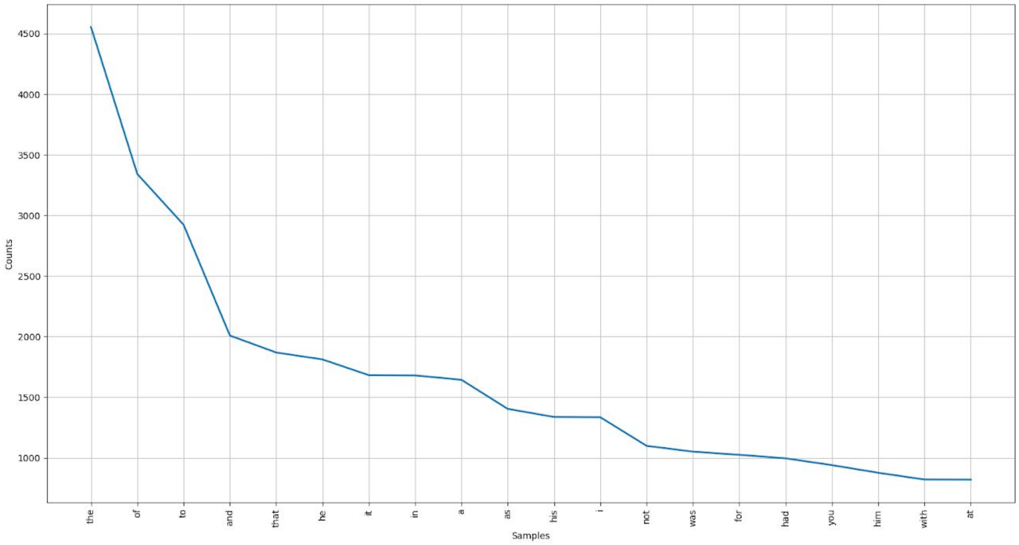
```
Extracting  http://www.gutenberg.org/files/63369/63369-0.txt
<FreqDist with 6472 samples and 68497 outcomes>
[('the', 4372), ('to', 1879), ('he', 1824), ('of', 1733), ('and', 1623), ('a', 1342), ('
“', 1135), ('his', 1134), ('”', 1120), ('was', 1082), ('in', 991), ('that', 948), ('it',
 873), ('’',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



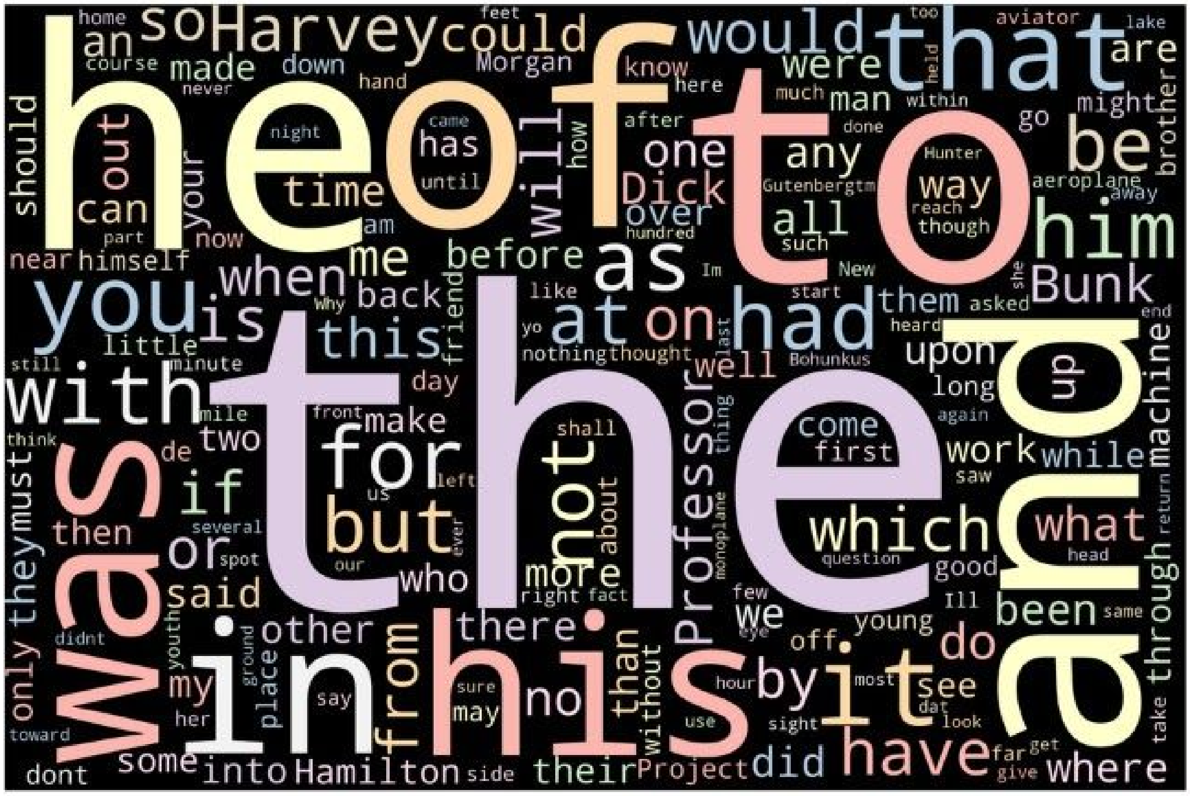
Problem statement - 5 :

Create a wordcloud before and after removing the stop words.

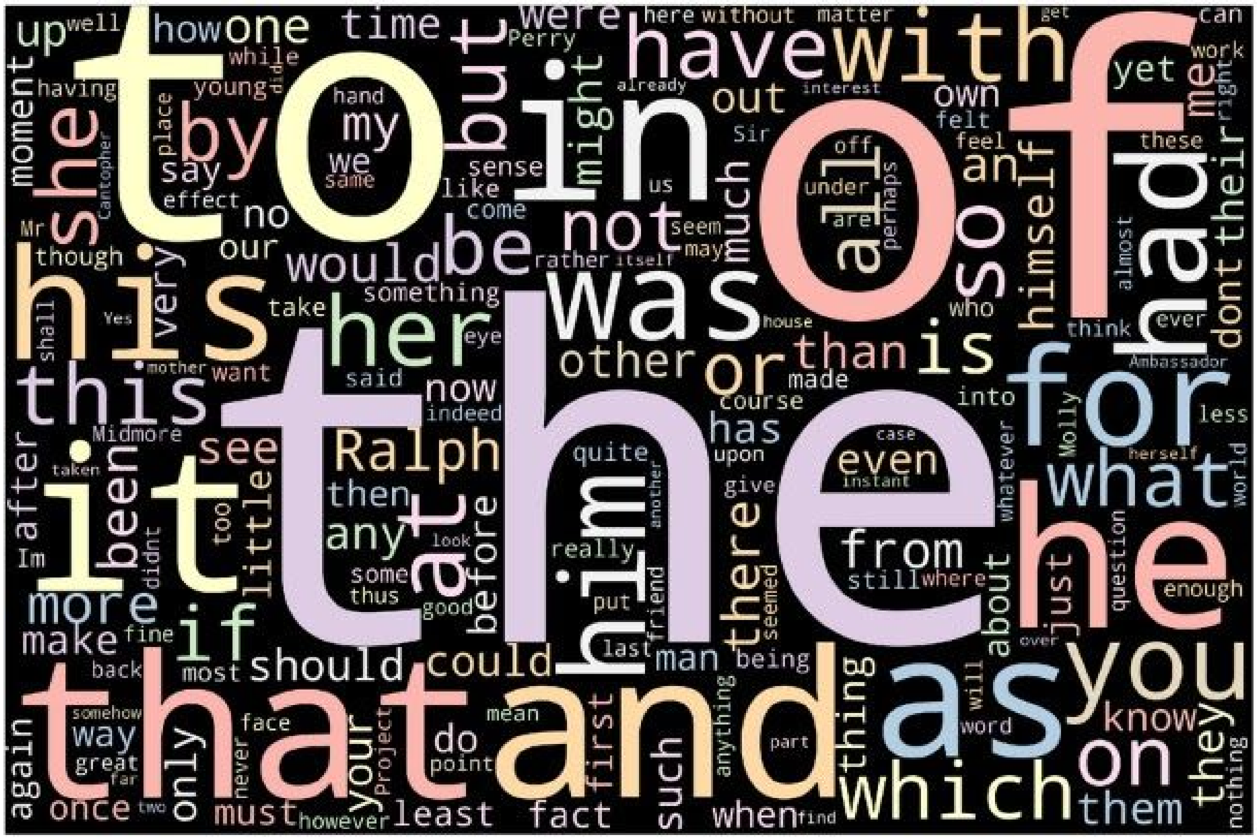
After analyzing the frequency distribution of the tokens, wordclouds are created using these tokens. A wordcloud is a visual representation of text data wherein each word is pictured with its frequency, i.e. higher the frequency, larger the size of the word. **Stopwords** - These are most commonly occurring words because of the grammar rules which hinder the visualization and useful computations.

Before removing stopwords :

Book-1 :



Book-2:

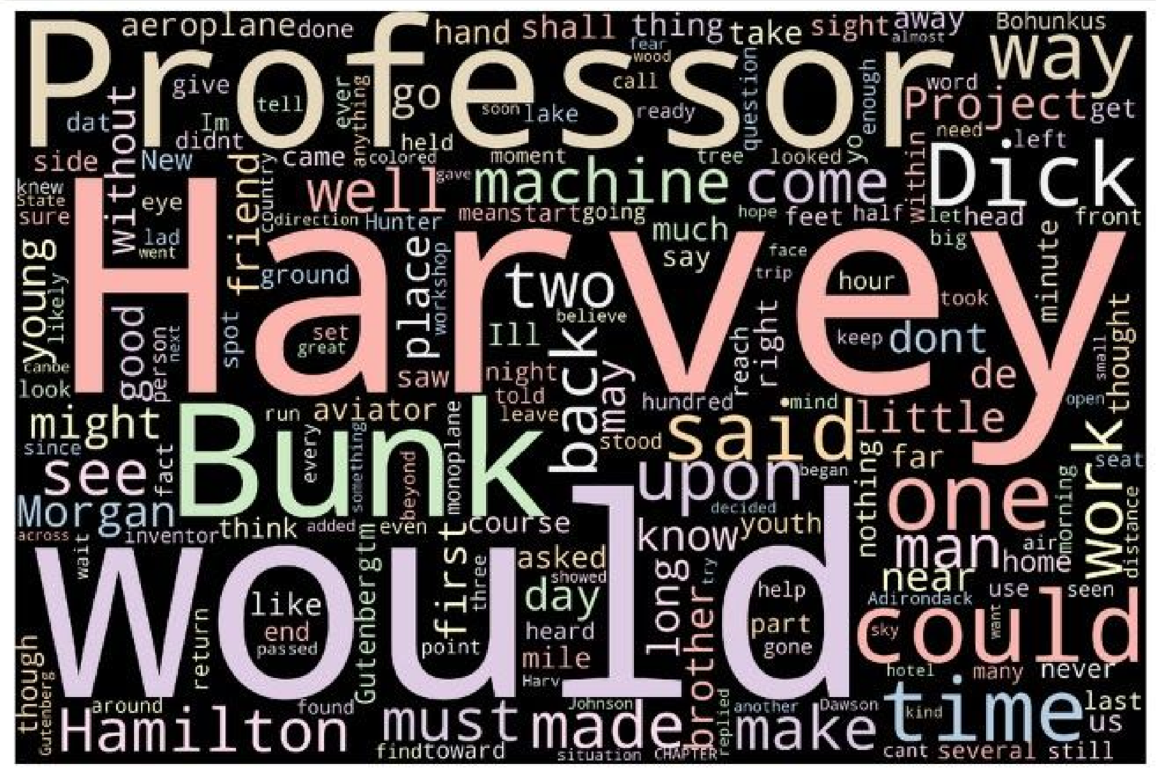


to
of
the
that
and
as
in
his
it
was
with
have
had
he
for
you
which
her
this
be
not
him
but
she
by
at
all
so
on
is
from
what
my
would
if
or
than
an
himself
there
thing
about
could
should
more
been
Ralph
other
one
how
up
well
time
were
here
without
matter
get
can
work
yet
me
right
their
dont
own
felt
feel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

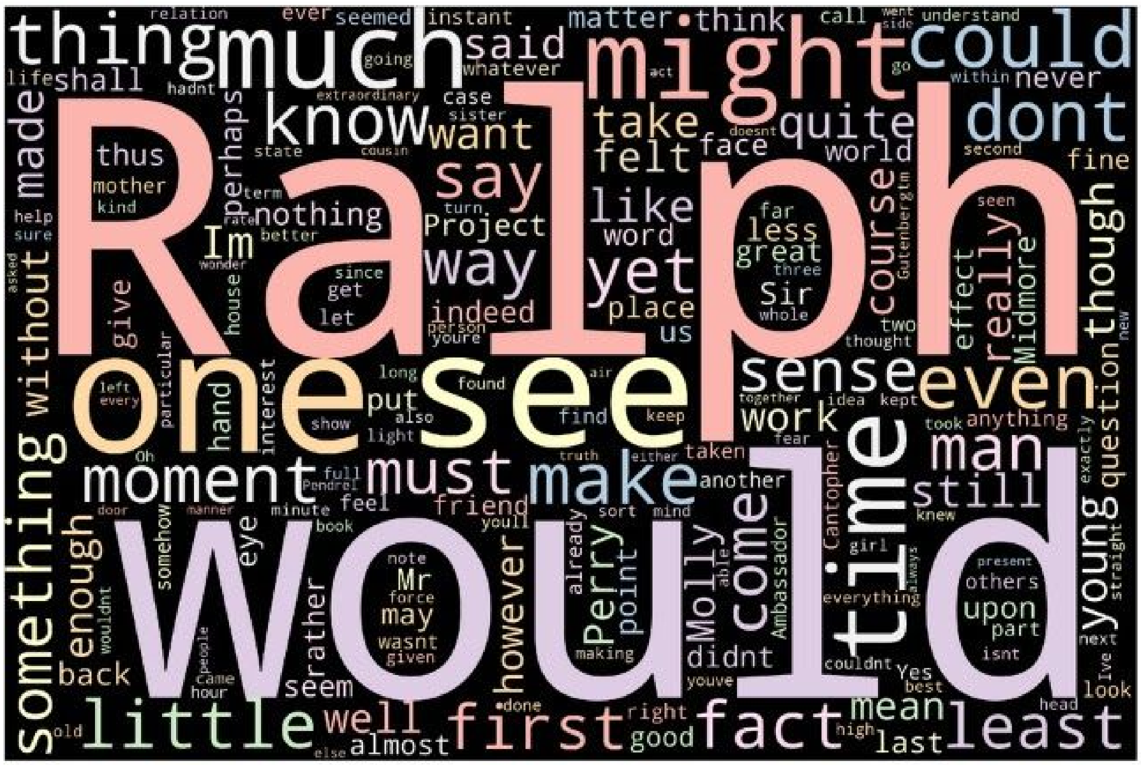
After removing stopwords :

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.

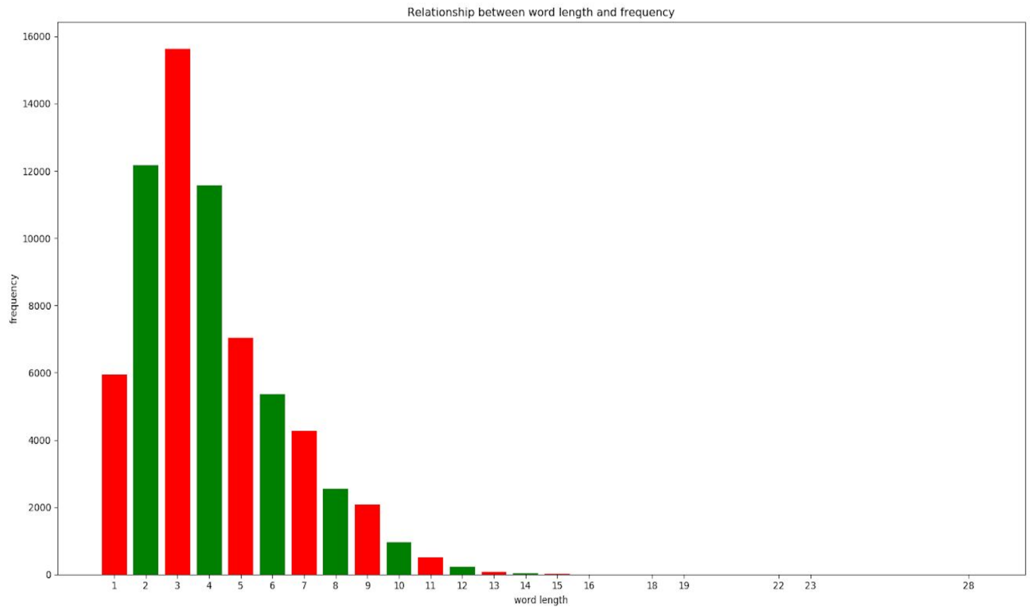
Problem statement - 6 :

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

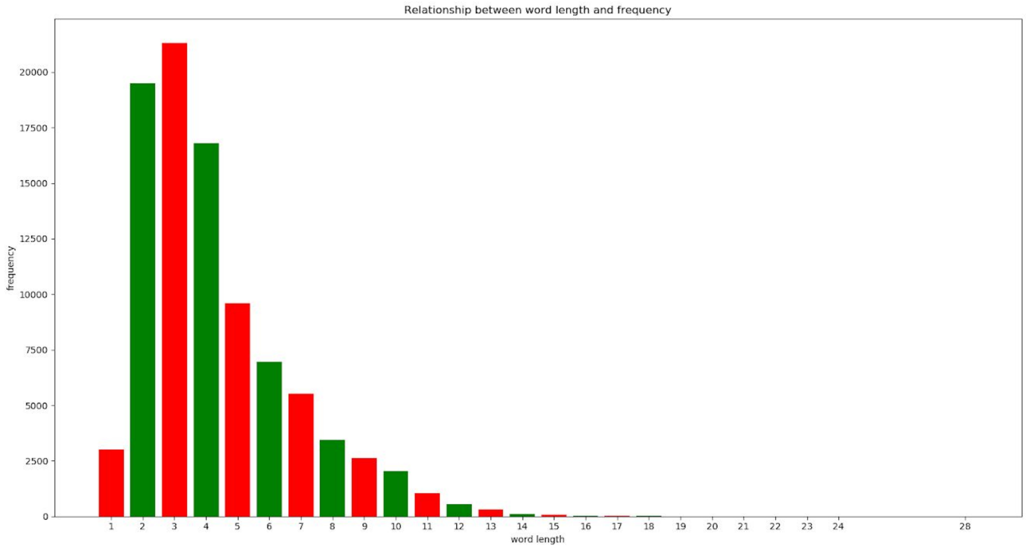
Firstly we created an empty list and stored the number of words for each length and then plotted it using Frequency Distribution and matplotlib.

Before Removing Stopwords:-

Book-1:

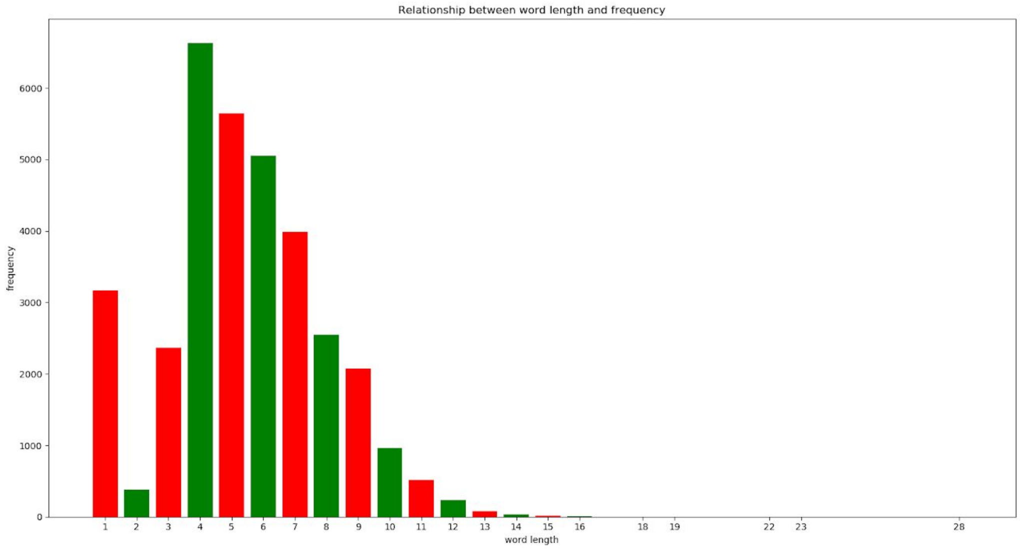


Book-2:

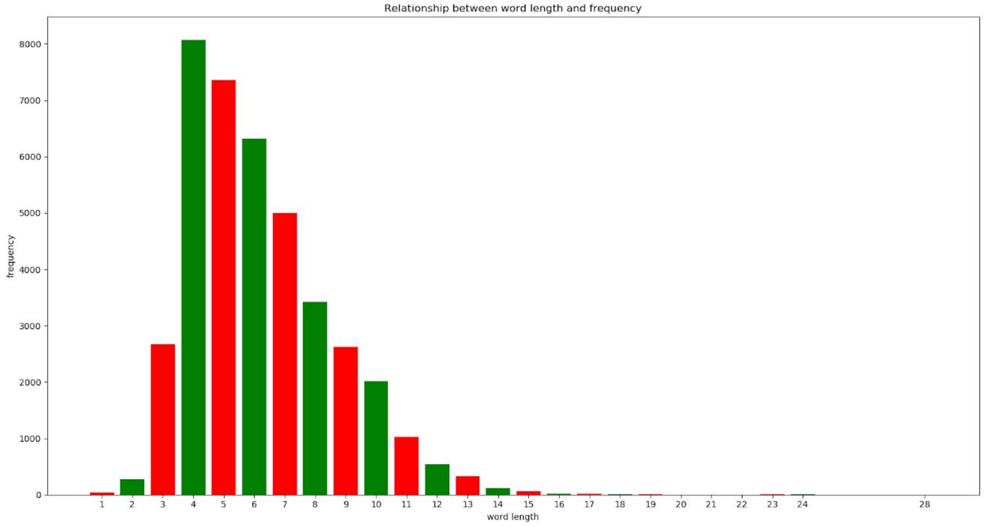


After Removing Stopwords:-

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 :

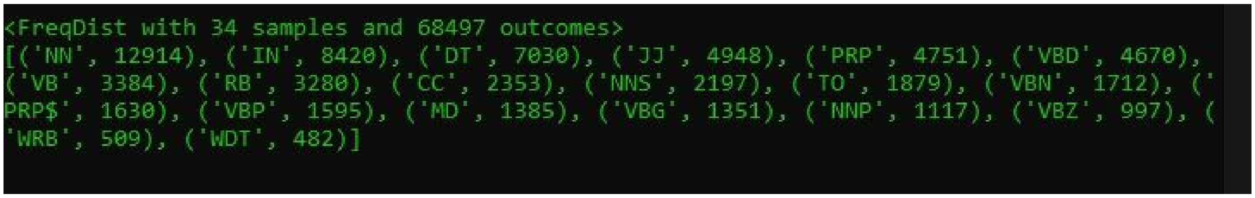
Do PoS Tagging using anyone of the four tagset studied in the class and get the distribution of various tags.

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.

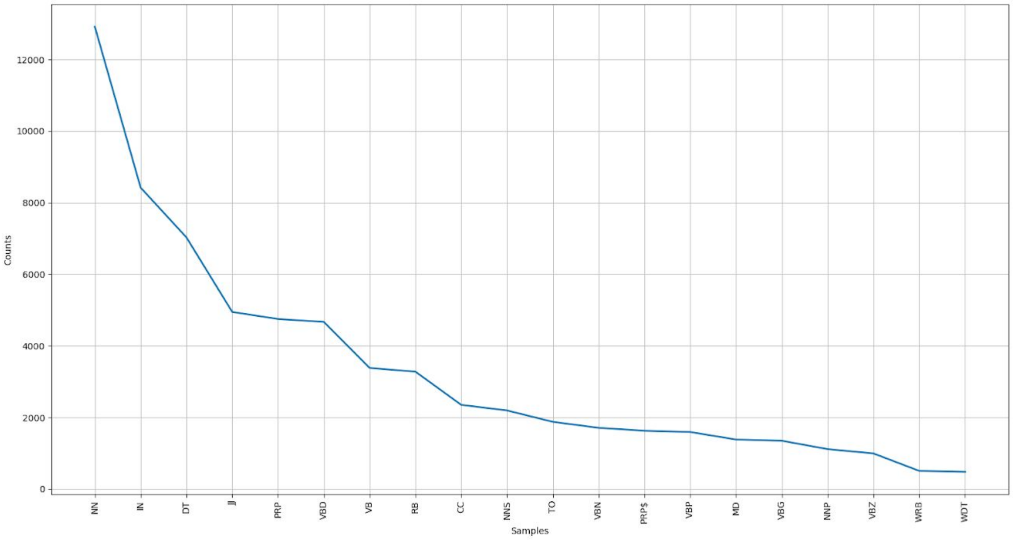
Book-1 :-

PoS tagging analysis



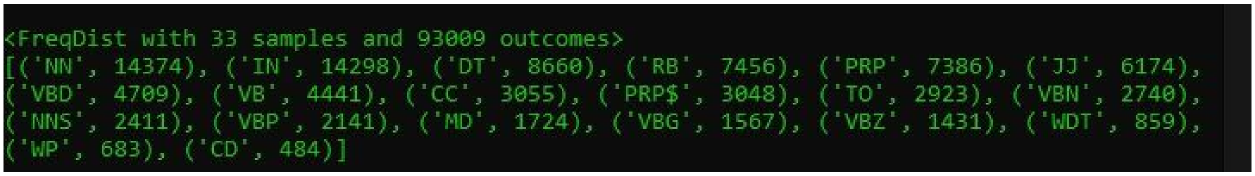
```
<FreqDist with 34 samples and 68497 outcomes>
[('NN', 12914), ('IN', 8420), ('DT', 7030), ('JJ', 4948), ('PRP', 4751), ('VBD', 4670),
('VB', 3384), ('RB', 3280), ('CC', 2353), ('NNS', 2197), ('TO', 1879), ('VBN', 1712), ('
PRP$', 1630), ('VBP', 1595), ('MD', 1385), ('VBG', 1351), ('NNP', 1117), ('VBZ',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

PoS tagging plot



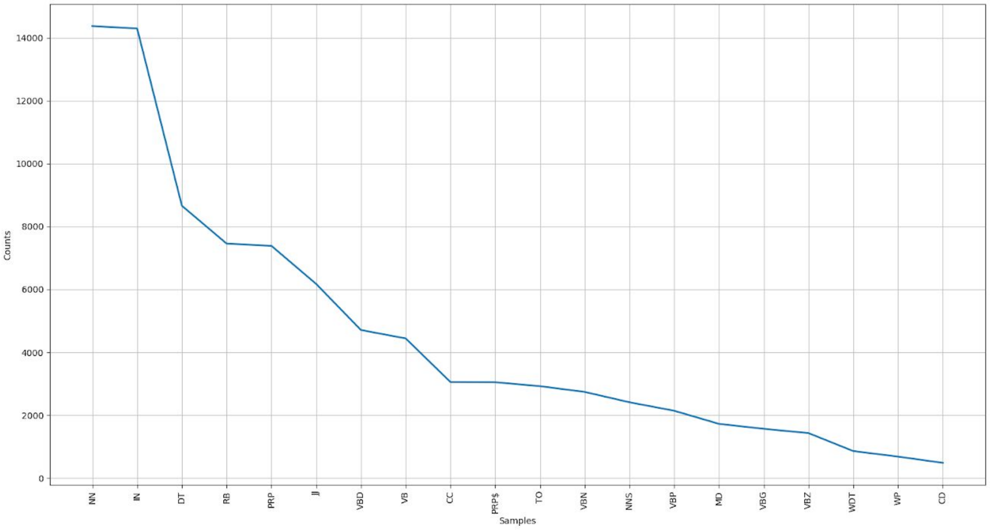
Book-2 :

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



Project Round-2

In this round we have used different books than the round 1 to analyze relations between entities in detailed manner.

Books Used:

Book-1 :

Title : The Mahabharata
Author : Kisari Mohan Ganguli
Produced by: David King, Juliet Sutherland
No. of words : *2,33,926*
No. of lines : *21,962*
No. of characters (without spaces) :*13,88,026*
Link : http://www.gutenberg.org//cache/epub/7864/pg7864.txt

Book-2 :

Title : Myths and Legends of Ancient Greece and Rome
Author : E.M. Berens
Illustrator : MAYNARD, MERRILL, & CO.
No. of words : *1,09,508*
No. of lines : *12,219*
No. of characters (without spaces) : *6,74,808*
Link : https://www.gutenberg.org//cache/epub/22381/pg22381.txt

Part - 1

Problem statement 1 :-

Find the nouns and verbs in both the novels. Get the categories that these words fall under in the WordNet. Note that there are 25 categories and 16 categories for Nouns and Verbs respectively. Get the frequency of each category for each noun and verb in their corresponding hierarchies and plot a histogram for the same for each novel.

- We first imported the books, preprocessed them, tokenized them, removed the stop words and then applied the PoS tagging to them (detailed explanation of these steps can be found above).

- Using PoS Tags, we extracted the nouns and verbs.

- For the extraction of nouns, we ran a for loop over the list of PoS tagged tuples which we got after PoS tagging. If the tag starts with ‘N’, then the corresponding word was appended in the nouns list. Similarly if the tag starts with ‘V’, the word belongs to the verbs list. Had the tag been directly compared with the string “NN”, some of the words having tags such as “NNP” would have been left out from our nouns list. Similar is the case with verbs.

- Now we have the nouns and verbs list. For each word in this list, its category was extracted from the word net.

WordNet :- It is the lexical database i.e. dictionary for different languages, specifically designed for natural language processing. For each word in the list, the list of its synsets is extracted using the function *wordnet.synsets(word)*.

Synsets :- It is a set of synonyms which share a common meaning. Moreover it is a simple interface that is present in NLTK to look up words in WordNet. Some of the words have only one Synset and some have several.

Now using the 1st element of this list of synsets, the category was extracted using the function *lexname().*

Noun categories present in the wordnet :-

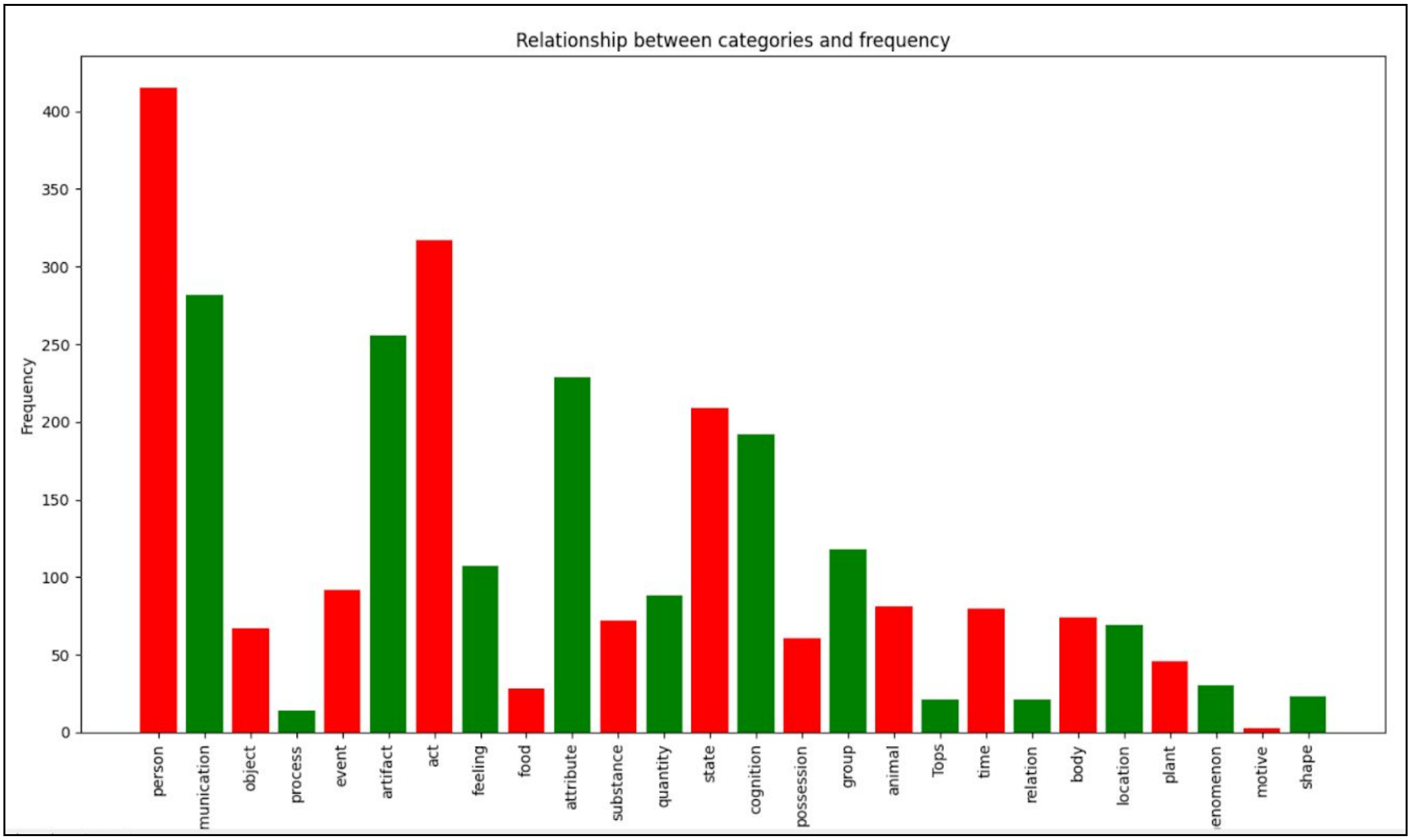
01	**tops**	unique beginner for nouns
02	**act**	nouns denoting acts or actions
03	**animal**	nouns denoting animals
04	**artifact**	nouns denoting man-made objects
05	**attribute**	nouns denoting attributes of people and objects
06	**body**	nouns denoting body parts
07	**cognition**	nouns denoting cognitive processes and contents
08	**communication**	nouns denoting communicative processes and contents
09	**event**	nouns denoting natural events

10	**feeling**	nouns denoting feelings and emotions
11	**food**	nouns denoting foods and drinks
12	**group**	nouns denoting groupings of people or objects
13	**location**	nouns denoting spatial position
14	**motive**	nouns denoting goals
15	**object**	nouns denoting natural objects (not man-made)
16	**person**	nouns denoting people
17	**phenomenon**	nouns denoting natural phenomena
18	**plant**	nouns denoting plants
19	**possession**	nouns denoting possession and transfer of possession
20	**process**	nouns denoting natural processes
21	**quantity**	nouns denoting quantities and units of measure
22	**relation**	nouns denoting relations between people or things or ideas
23	**shape**	nouns denoting two and three dimensional shapes
24	**state**	nouns denoting stable states of affairs
25	**substance**	nouns denoting substances
26	**time**	nouns denoting time and temporal relations

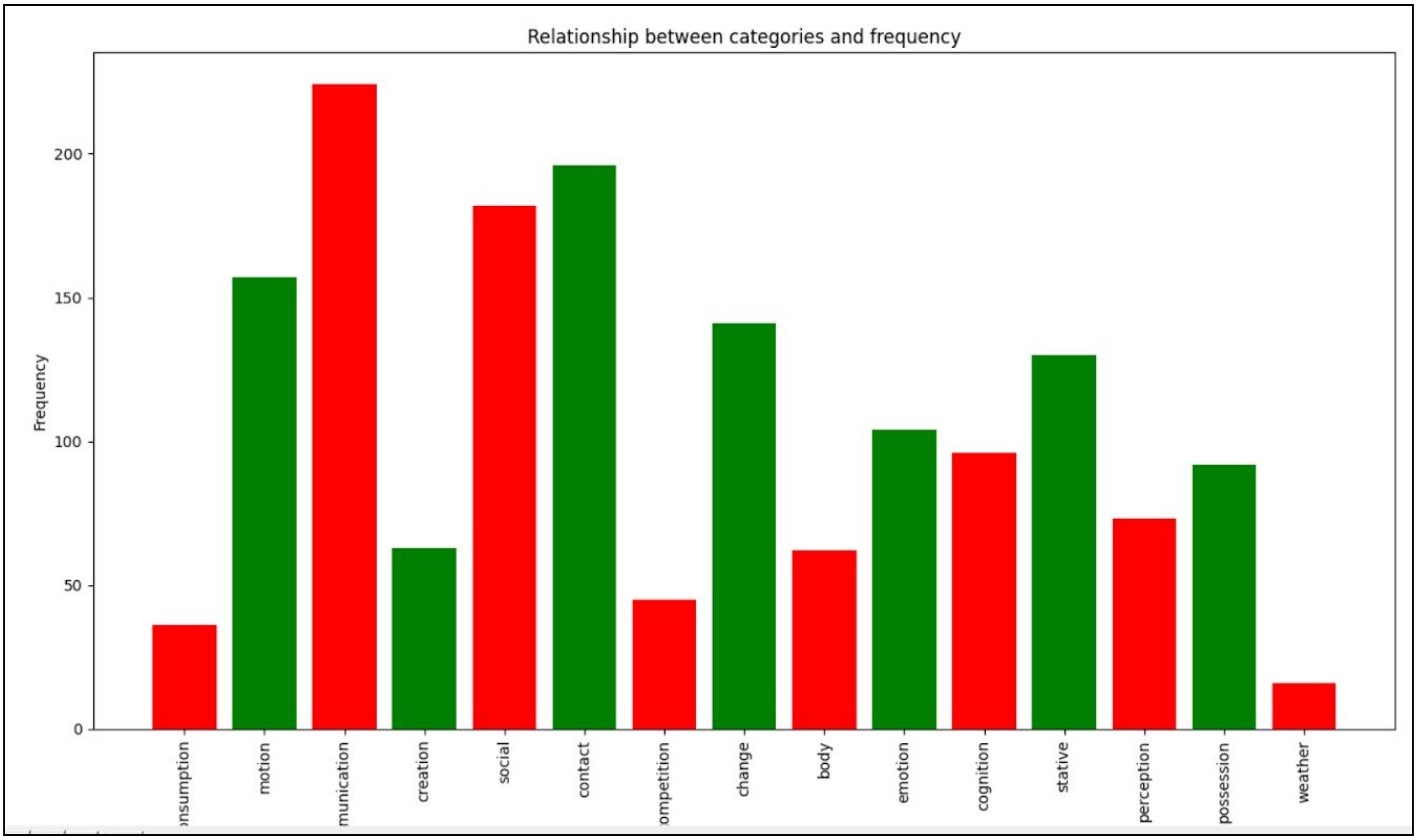
Verb categories present in the wordnet :-

1	**body**	verbs of grooming, dressing and bodily care
2	**change**	verbs of size, temperature change, intensifying, etc.
3	**cognition**	verbs of thinking, judging, analyzing, doubting
4	**communication**	verbs of telling, asking, ordering, singing
5	**competition**	verbs of fighting, athletic activities
6	**consumption**	verbs of eating and drinking
7	**contact**	verbs of touching, hitting, tying, digging
8	**creation**	verbs of sewing, baking, painting, performing
9	**emotion**	verbs of feeling
10	**motion**	verbs of walking, flying, swimming
11	**perception**	verbs of seeing, hearing, feeling
12	**possession**	verbs of buying, selling, owning
13	**social**	verbs of political and social activities and events
14	**stative**	verbs of being, having, spatial relations
15	**weather**	verbs of raining, snowing, thawing, thundering

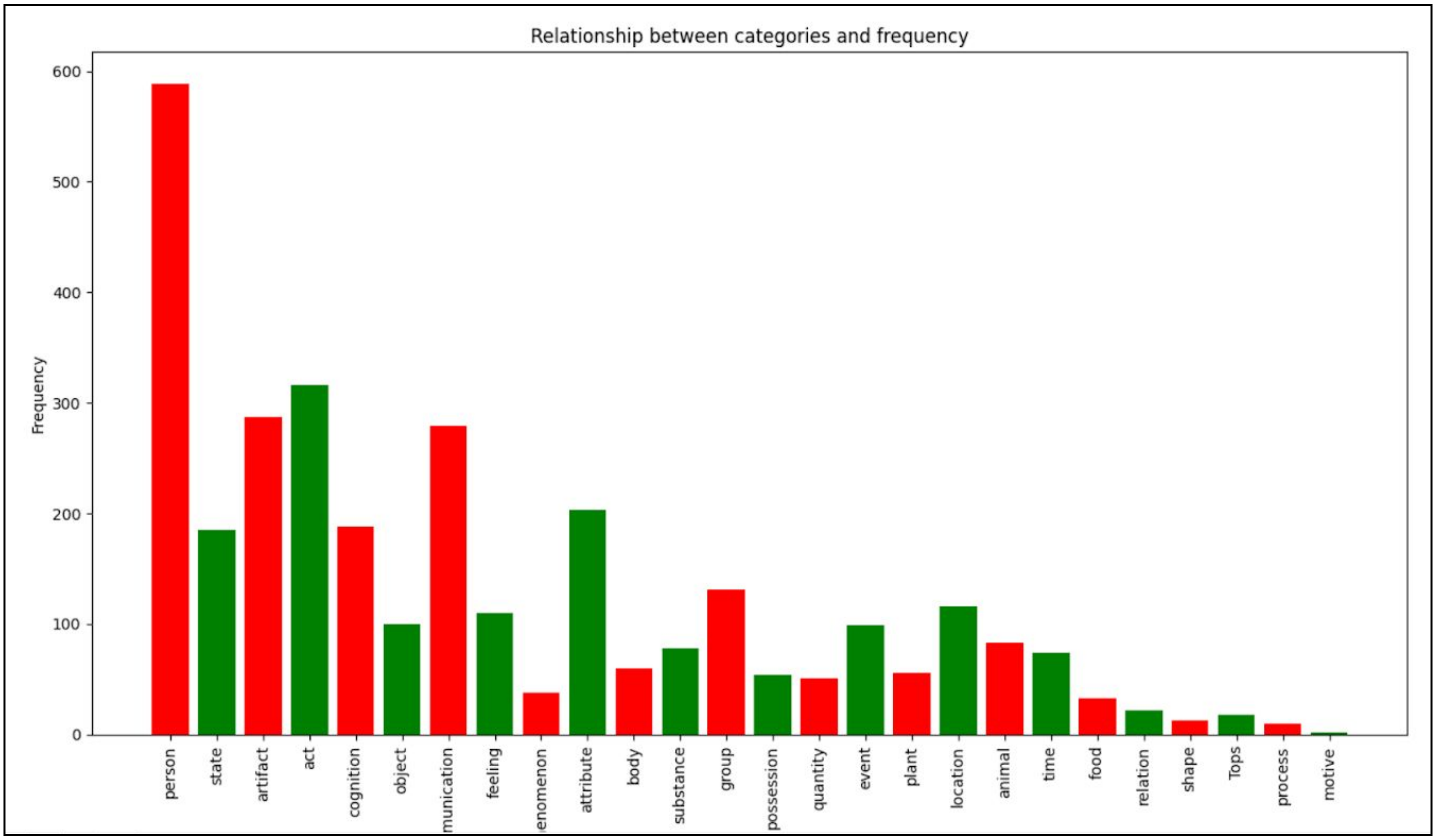
Frequency plot of the noun categories of the **nouns** of **book-1**



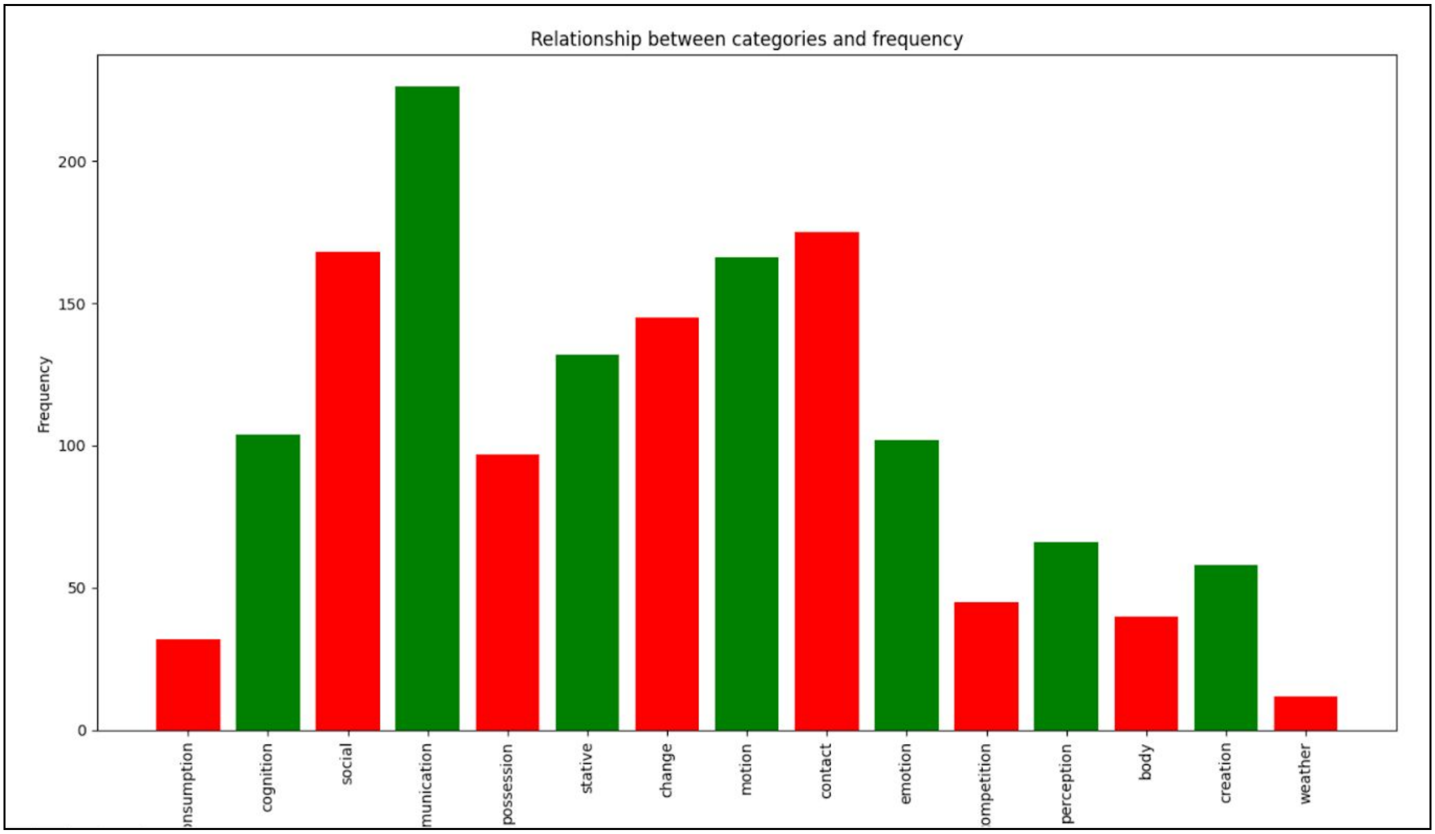
Frequency plot of the verb categories of the **verbs** of **book-1**



Frequency plot of the noun categories of the **nouns** of **book-2**



Frequency plot of the verb categories of the **verbs** of **book-2**



Part-2

Problem Statement :-

Recognise all the named entities in both the books.

Steps :

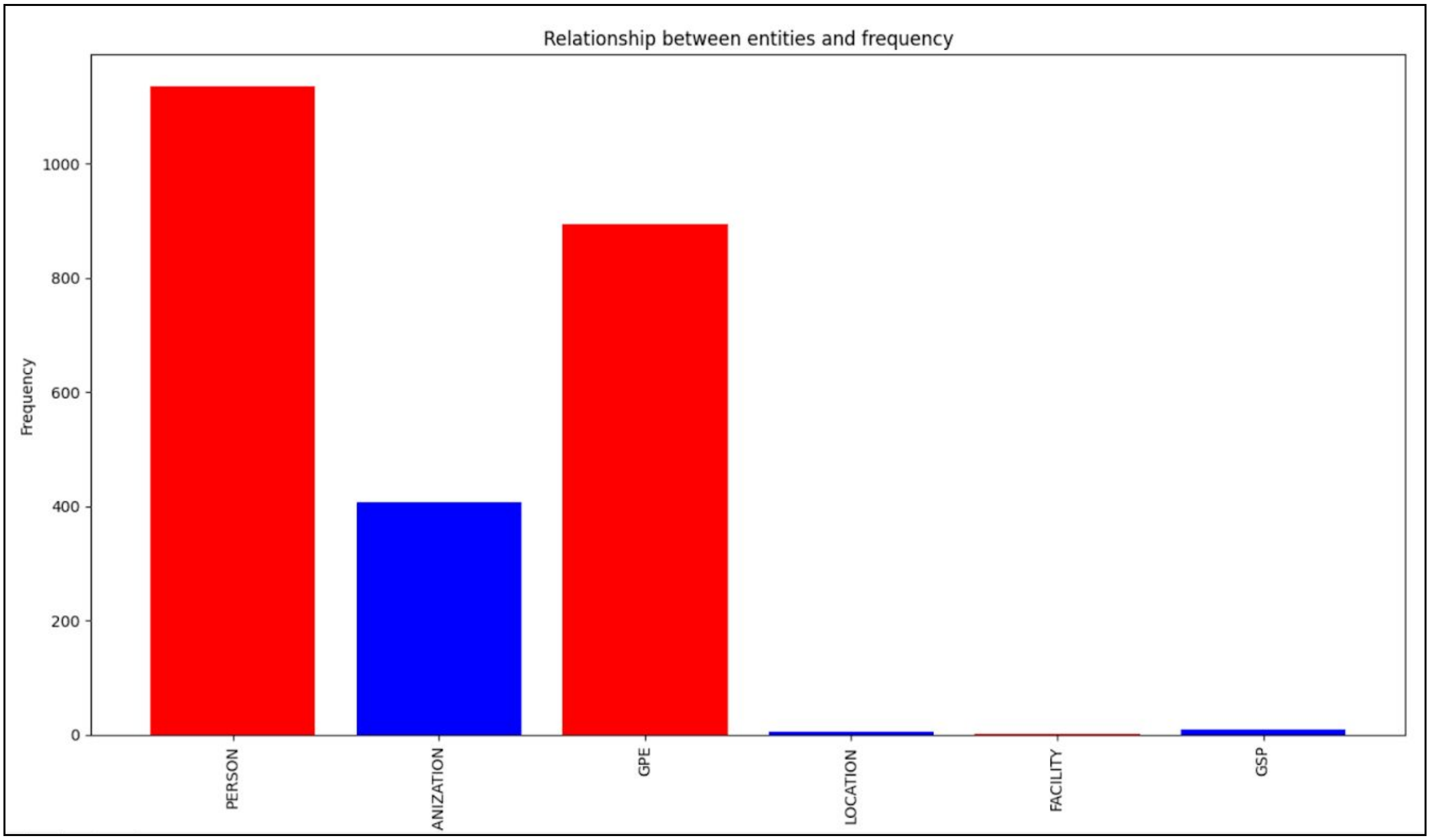
- First without doing any preprocessing, we perform tokenization for both the books.
- Both the books are POS tagged using nltk library.
- Now, on this tagged data, we perform entity recognition using *ne_chunk()* function.
- All the used entities are recorded for both the books.

Entity Recognition :

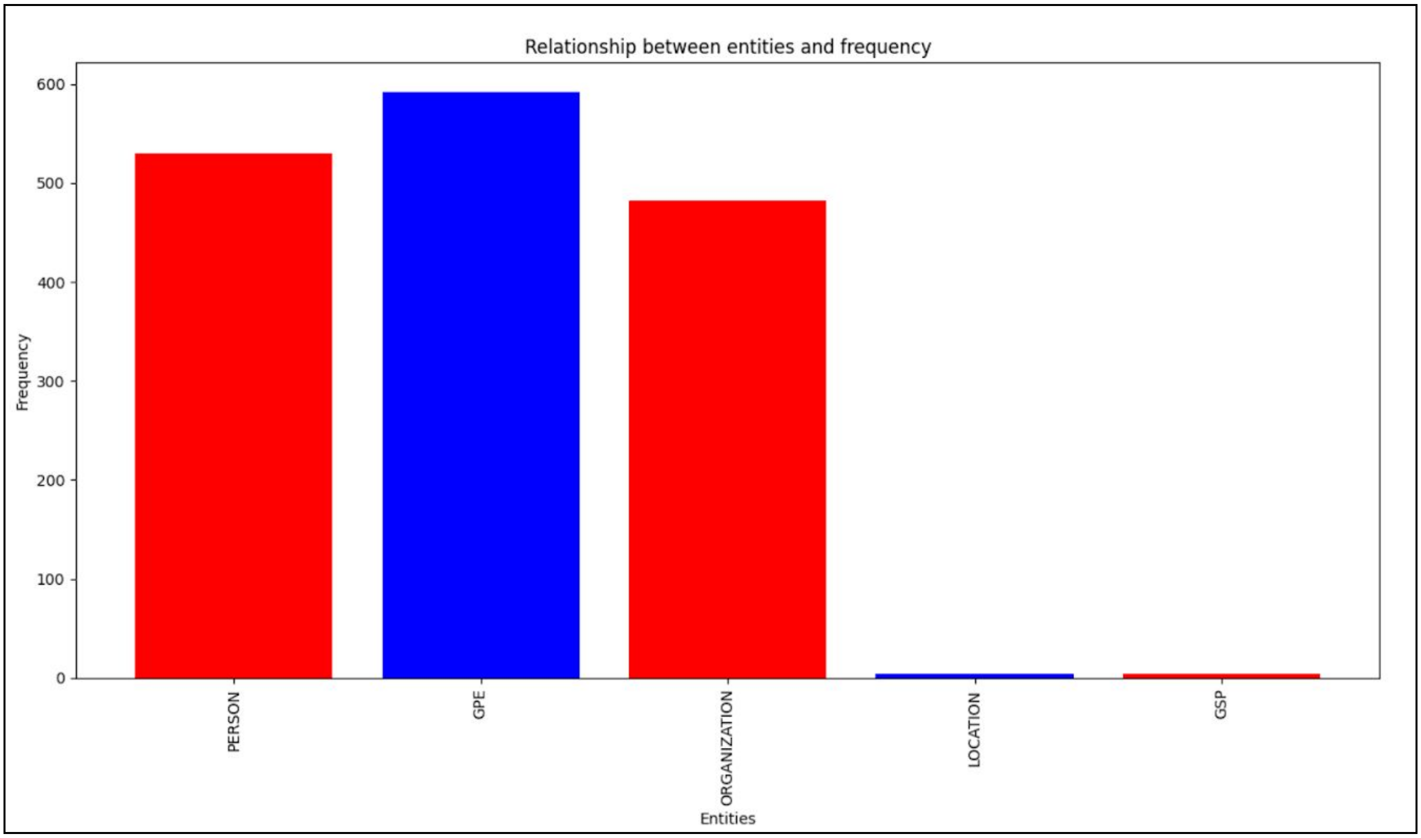
- Sequence labelling is used to perform Named Entity Recognition (NER).
- First we encode our training data with IOB tags. This is done manually by domain experts.
- Set of features are associated with each token to be labelled.
- When an adequate set of features are extracted from a training set, it is encoded in a form appropriate to train a machine learning based sequence classifier.
- These extracted features are augmented with our earlier IOB scheme with more columns.
- Now a sequence classifier can be trained.

VISUALIZATION :-

BOOK 1:



BOOK 2:



Performance Evaluation :

- We need a perfectly labeled data to compare our classifier with.
- We selected two random passages from each text book and manually tagged it ourselves with the right entity. For this purpose we used a particular entity, i.e., PER (person).
- Then we used our classifier to tag these passages.
- We created a function *(metrics)* to get the True Positives, False Negatives, False positives and True negatives.
- Using these values we found out the accuracy, precision, recall and f-measure of our data.

Result :

Two lists were made, namely “*truth”* and “*run”*. The *truth* list contains the manually labelled entities and *run* list contains the entities that were recognized by the algorithm. (The entity used for this purpose is Person).

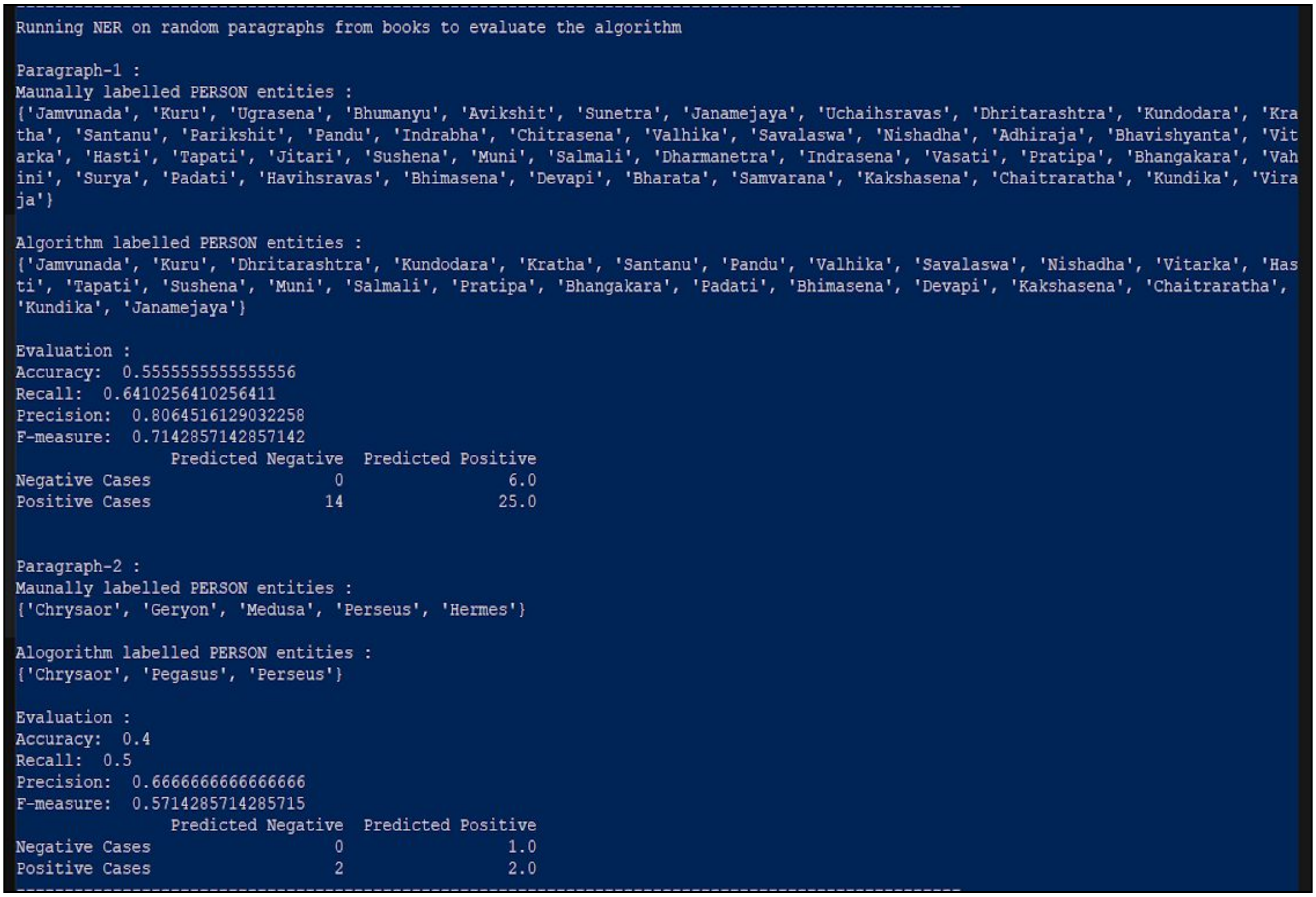
The confusion matrix was made :-

	PREDICTED CLASS		
ACTUAL CLASS		**Yes**	**No**
	Yes	***TP***	***FP***
	No	***TN***	***FN***

TP refers to true positive, FP refers to false positive, FN refers to false negative and TN refers to true negative. TP is the count of positive records which are classified as positive. FP is the count of positive records which are not classified as positive. TN is the count of negative records which are classified as negative. FN is the count of negative records which are not classified as negative.

These values were calculated from the *truth* and *run* lists. Using these values, *accuracy*, *recall*, *precision* and *F-measure* are calculated via following formulas :

Accuracy = (TP + TN) / (TP + TN + FP + FN)
Recall = TP / (TP + FP)
Precision = TP / (TP + TN)
F-measure = (2 * recall * precision) / (recall + precision)



```
----------------------------------------------------------------------------------------------------
Running NER on random paragraphs from books to evaluate the algorithm

Paragraph-1 :
Maunally labelled PERSON entities :
{'Jamvunada', 'Kuru', 'Ugrasena', 'Bhumanyu', 'Avikshit', 'Sunetra', 'Janamejaya', 'Uchaihsravas', 'Dhritarashtra', 'Kundodara', 'Kra
tha', 'Santanu', 'Parikshit', 'Pandu', 'Indrabha', 'Chitrasena', 'Valhika', 'Savalaswa', 'Nishadha', 'Adhiraja', 'Bhavishyanta', 'Vit
arka', 'Hasti', 'Tapati', 'Jitari', 'Sushena', 'Muni', 'Salmali', 'Dharmanetra', 'Indrasena', 'Vasati', 'Pratipa', 'Bhangakara', 'Vah
ini', 'Surya', 'Padati', 'Havihsravas', 'Bhimasena', 'Devapi', 'Bharata', 'Samvarana', 'Kakshasena', 'Chaitraratha', 'Kundika', 'Vira
ja'}

Algorithm labelled PERSON entities :
{'Jamvunada', 'Kuru', 'Dhritarashtra', 'Kundodara', 'Kratha', 'Santanu', 'Pandu', 'Valhika', 'Savalaswa', 'Nishadha', 'Vitarka', 'Has
ti', 'Tapati', 'Sushena', 'Muni', 'Salmali', 'Pratipa', 'Bhangakara', 'Padati', 'Bhimasena', 'Devapi', 'Kakshasena', 'Chaitraratha',
'Kundika', 'Janamejaya'}

Evaluation :
Accuracy:  0.5555555555555556
Recall:  0.6410256410256411
Precision:  0.8064516129032258
F-measure:  0.7142857142857142
                Predicted Negative  Predicted Positive
Negative Cases                   0                 6.0
Positive Cases                  14                25.0


Paragraph-2 :
Maunally labelled PERSON entities :
{'Chrysaor', 'Geryon', 'Medusa', 'Perseus', 'Hermes'}

Alogorithm labelled PERSON entities :
{'Chrysaor', 'Pegasus', 'Perseus'}

Evaluation :
Accuracy:  0.4
Recall:  0.5
Precision:  0.6666666666666666
F-measure:  0.5714285714285715
                Predicted Negative  Predicted Positive
Negative Cases                   0                 1.0
Positive Cases                   2                 2.0
----------------------------------------------------------------------------------------------------
```

Part-3

Problem Statement :-

Extract the relationship between the entities (mainly the characters involved in the novel).

We first imported the books, tokenized the texts, applied PoS tagging and then applied the named entity recognition as done in part 2 to get the entity chunks.

Relationship extraction :-

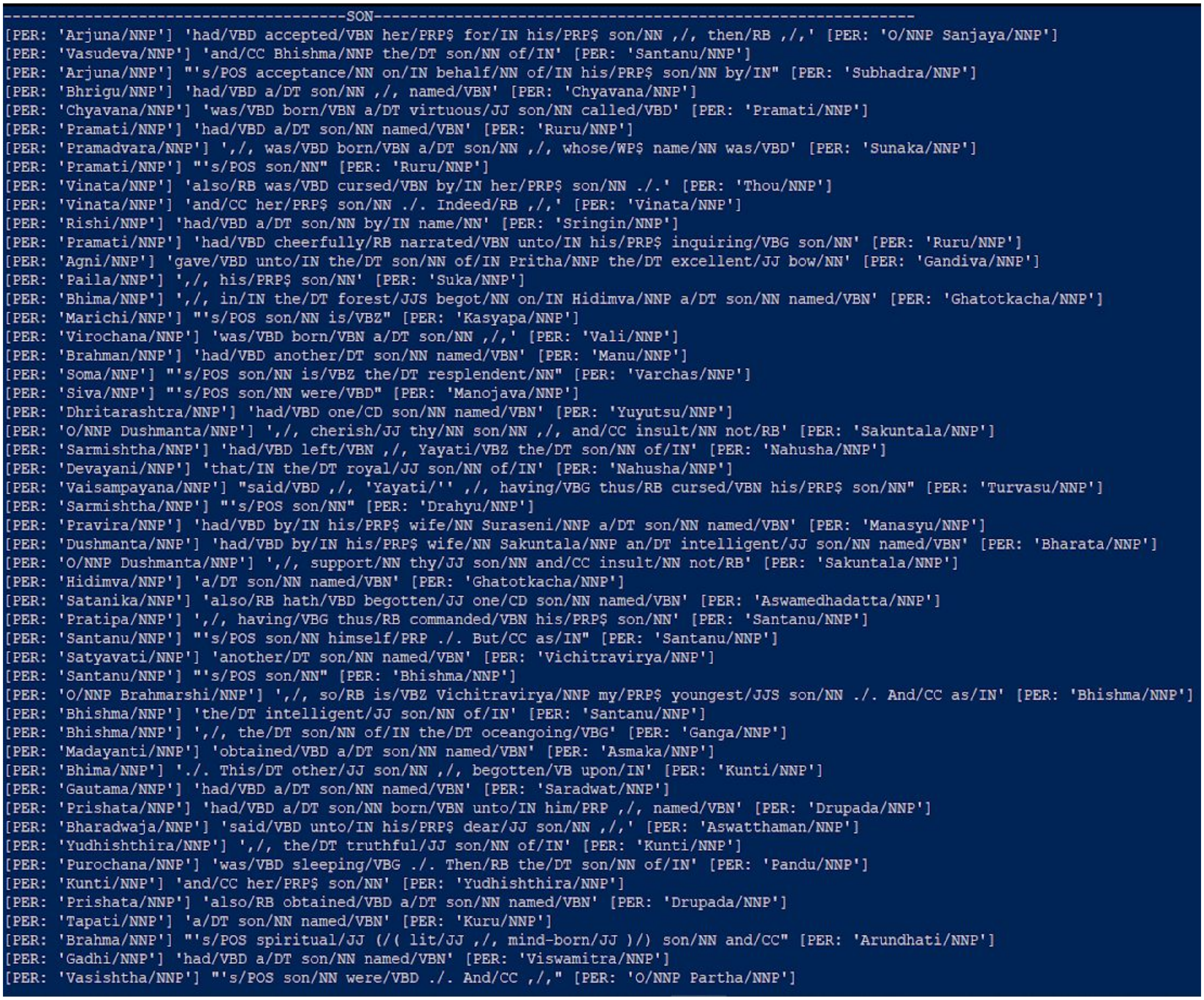
- For relationship extraction we are using regular expressions, to find patterns that match text segments which are likely to contain expressions of relations we are interested in.
- We generated certain regular expressions which clearly identifies PERSON to PERSON relations.
- For example, take the sentence, Ram is the son of Shyam, which is already entity tagged i.e Ram: Person and Shyam: Person, by applying the Regular expression *\bson\b* on this sentence, we will get a relationship as:

  PER{Ram} son of PER{Ram}
- We have found relations of *'son of'* , *'brother of'*, *'daughter of'* for both the books.

Result :-

Book-1

Son :-

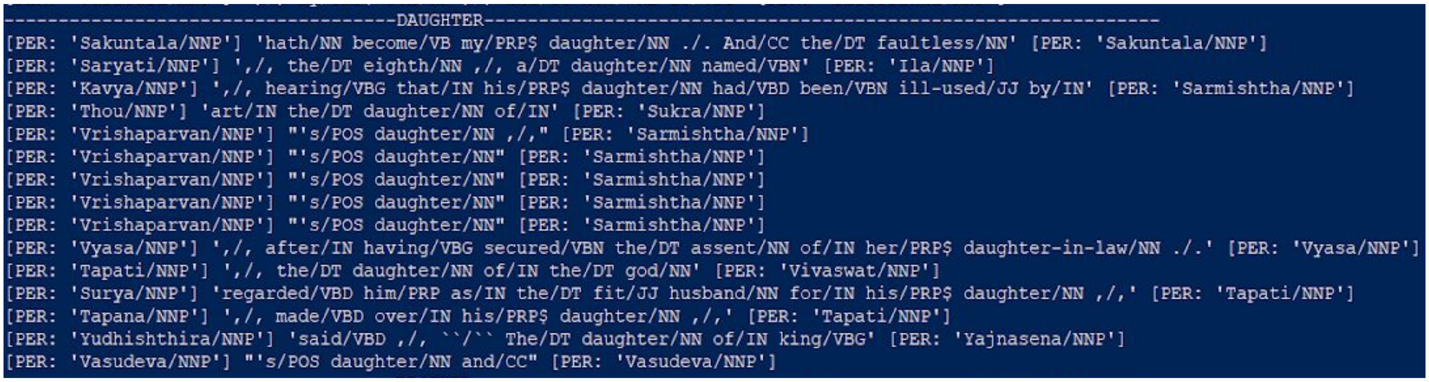


```
--------------------------------------SON------------------------------------------------------------
[PER: 'Arjuna/NNP'] 'had/VBD accepted/VBN her/PRP$ for/IN his/PRP$ son/NN ,/, then/RB ,/,' [PER: 'O/NNP Sanjaya/NNP']
[PER: 'Vasudeva/NNP'] 'and/CC Bhishma/NNP the/DT son/NN of/IN' [PER: 'Santanu/NNP']
[PER: 'Arjuna/NNP'] "'s/POS acceptance/NN on/IN behalf/NN of/IN his/PRP$ son/NN by/IN" [PER: 'Subhadra/NNP']
[PER: 'Bhrigu/NNP'] 'had/VBD a/DT son/NN ,/, named/VBN' [PER: 'Chyavana/NNP']
[PER: 'Chyavana/NNP'] 'was/VBD born/VBN a/DT virtuous/JJ son/NN called/VBD' [PER: 'Pramati/NNP']
[PER: 'Pramati/NNP'] 'had/VBD a/DT son/NN named/VBN' [PER: 'Ruru/NNP']
[PER: 'Pramadvara/NNP'] ',/, was/VBD born/VBN a/DT son/NN ,/, whose/WP$ name/NN was/VBD' [PER: 'Sunaka/NNP']
[PER: 'Pramati/NNP'] "'s/POS son/NN" [PER: 'Ruru/NNP']
[PER: 'Vinata/NNP'] 'also/RB was/VBD cursed/VBN by/IN her/PRP$ son/NN ./.' [PER: 'Thou/NNP']
[PER: 'Vinata/NNP'] 'and/CC her/PRP$ son/NN ./. Indeed/RB ,/,' [PER: 'Vinata/NNP']
[PER: 'Rishi/NNP'] 'had/VBD a/DT son/NN by/IN name/NN' [PER: 'Sringin/NNP']
[PER: 'Pramati/NNP'] 'had/VBD cheerfully/RB narrated/VBN unto/IN his/PRP$ inquiring/VBG son/NN' [PER: 'Ruru/NNP']
[PER: 'Agni/NNP'] 'gave/VBD unto/IN the/DT son/NN of/IN Pritha/NNP the/DT excellent/JJ bow/NN' [PER: 'Gandiva/NNP']
[PER: 'Paila/NNP'] ',/, his/PRP$ son/NN' [PER: 'Suka/NNP']
[PER: 'Bhima/NNP'] ',/, in/IN the/DT forest/JJS begot/NN on/IN Hidimva/NNP a/DT son/NN named/VBN' [PER: 'Ghatotkacha/NNP']
[PER: 'Marichi/NNP'] "'s/POS son/NN is/VBZ" [PER: 'Kasyapa/NNP']
[PER: 'Virochana/NNP'] 'was/VBD born/VBN a/DT son/NN ,/,' [PER: 'Vali/NNP']
[PER: 'Brahman/NNP'] 'had/VBD another/DT son/NN named/VBN' [PER: 'Manu/NNP']
[PER: 'Soma/NNP'] "'s/POS son/NN is/VBZ the/DT resplendent/NN" [PER: 'Varchas/NNP']
[PER: 'Siva/NNP'] "'s/POS son/NN were/VBD" [PER: 'Manojava/NNP']
[PER: 'Dhritarashtra/NNP'] 'had/VBD one/CD son/NN named/VBN' [PER: 'Yuyutsu/NNP']
[PER: 'O/NNP Dushmanta/NNP'] ',/, cherish/JJ thy/NN son/NN ,/, and/CC insult/NN not/RB' [PER: 'Sakuntala/NNP']
[PER: 'Sarmishtha/NNP'] 'had/VBD left/VBN ,/, Yayati/VBZ the/DT son/NN of/IN' [PER: 'Nahusha/NNP']
[PER: 'Devayani/NNP'] 'that/IN the/DT royal/JJ son/NN of/IN' [PER: 'Nahusha/NNP']
[PER: 'Vaisampayana/NNP'] "said/VBD ,/, 'Yayati/'' ,/, having/VBG thus/RB cursed/VBN his/PRP$ son/NN" [PER: 'Turvasu/NNP']
[PER: 'Sarmishtha/NNP'] "'s/POS son/NN" [PER: 'Drahyu/NNP']
[PER: 'Pravira/NNP'] 'had/VBD by/IN his/PRP$ wife/NN Suraseni/NNP a/DT son/NN named/VBN' [PER: 'Manasyu/NNP']
[PER: 'Dushmanta/NNP'] 'had/VBD by/IN his/PRP$ wife/NN Sakuntala/NNP an/DT intelligent/JJ son/NN named/VBN' [PER: 'Bharata/NNP']
[PER: 'O/NNP Dushmanta/NNP'] ',/, support/NN thy/JJ son/NN and/CC insult/NN not/RB' [PER: 'Sakuntala/NNP']
[PER: 'Hidimva/NNP'] 'a/DT son/NN named/VBN' [PER: 'Ghatotkacha/NNP']
[PER: 'Satanika/NNP'] 'also/RB hath/VBD begotten/JJ one/CD son/NN named/VBN' [PER: 'Aswamedhadatta/NNP']
[PER: 'Pratipa/NNP'] ',/, having/VBG thus/RB commanded/VBN his/PRP$ son/NN' [PER: 'Santanu/NNP']
[PER: 'Santanu/NNP'] "'s/POS son/NN himself/PRP ./. But/CC as/IN" [PER: 'Santanu/NNP']
[PER: 'Satyavati/NNP'] 'another/DT son/NN named/VBN' [PER: 'Vichitravirya/NNP']
[PER: 'Santanu/NNP'] "'s/POS son/NN" [PER: 'Bhishma/NNP']
[PER: 'O/NNP Brahmarshi/NNP'] ',/, so/RB is/VBZ Vichitravirya/NNP my/PRP$ youngest/JJS son/NN ./. And/CC as/IN' [PER: 'Bhishma/NNP']
[PER: 'Bhishma/NNP'] 'the/DT intelligent/JJ son/NN of/IN' [PER: 'Santanu/NNP']
[PER: 'Bhishma/NNP'] ',/, the/DT son/NN of/IN the/DT oceangoing/VBG' [PER: 'Ganga/NNP']
[PER: 'Madayanti/NNP'] 'obtained/VBD a/DT son/NN named/VBN' [PER: 'Asmaka/NNP']
[PER: 'Bhima/NNP'] './. This/DT other/JJ son/NN ,/, begotten/VB upon/IN' [PER: 'Kunti/NNP']
[PER: 'Gautama/NNP'] 'had/VBD a/DT son/NN named/VBN' [PER: 'Saradwat/NNP']
[PER: 'Prishata/NNP'] 'had/VBD a/DT son/NN born/VBN unto/IN him/PRP ,/, named/VBN' [PER: 'Drupada/NNP']
[PER: 'Bharadwaja/NNP'] 'said/VBD unto/IN his/PRP$ dear/JJ son/NN ,/,' [PER: 'Aswatthaman/NNP']
[PER: 'Yudhishthira/NNP'] ',/, the/DT truthful/JJ son/NN of/IN' [PER: 'Kunti/NNP']
[PER: 'Purochana/NNP'] 'was/VBD sleeping/VBG ./. Then/RB the/DT son/NN of/IN' [PER: 'Pandu/NNP']
[PER: 'Kunti/NNP'] 'and/CC her/PRP$ son/NN' [PER: 'Yudhishthira/NNP']
[PER: 'Prishata/NNP'] 'also/RB obtained/VBD a/DT son/NN named/VBN' [PER: 'Drupada/NNP']
[PER: 'Tapati/NNP'] 'a/DT son/NN named/VBN' [PER: 'Kuru/NNP']
[PER: 'Brahma/NNP'] "'s/POS spiritual/JJ (/( lit/JJ ,/, mind-born/JJ )/) son/NN and/CC" [PER: 'Arundhati/NNP']
[PER: 'Gadhi/NNP'] 'had/VBD a/DT son/NN named/VBN' [PER: 'Viswamitra/NNP']
[PER: 'Vasishtha/NNP'] "'s/POS son/NN were/VBD ./. And/CC ,/," [PER: 'O/NNP Partha/NNP']
```

Some of the relations drawn from above:-

- ***Saradwat*** is the son of ***Gautam***.
- ***Soma*** is the son of ***Varchas***.
- ***Drupada*** is the son of ***Prishata***.
- ***Yuyutu*** is the son of ***Dhritrashtra*** .

Daughter :-

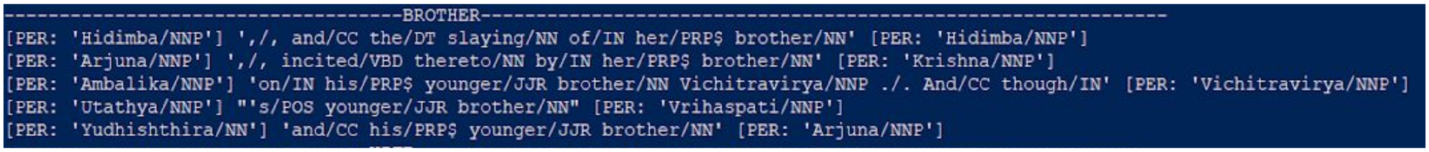


```
------------------------------------DAUGHTER---------------------------------------------------------------
[PER: 'Sakuntala/NNP'] 'hath/NN become/VB my/PRP$ daughter/NN ./. And/CC the/DT faultless/NN' [PER: 'Sakuntala/NNP']
[PER: 'Saryati/NNP'] ',/, the/DT eighth/NN ,/, a/DT daughter/NN named/VBN' [PER: 'Ila/NNP']
[PER: 'Kavya/NNP'] ',/, hearing/VBG that/IN his/PRP$ daughter/NN had/VBD been/VBN ill-used/JJ by/IN' [PER: 'Sarmishtha/NNP']
[PER: 'Thou/NNP'] 'art/IN the/DT daughter/NN of/IN' [PER: 'Sukra/NNP']
[PER: 'Vrishaparvan/NNP'] "'s/POS daughter/NN ,/," [PER: 'Sarmishtha/NNP']
[PER: 'Vrishaparvan/NNP'] "'s/POS daughter/NN" [PER: 'Sarmishtha/NNP']
[PER: 'Vrishaparvan/NNP'] "'s/POS daughter/NN" [PER: 'Sarmishtha/NNP']
[PER: 'Vrishaparvan/NNP'] "'s/POS daughter/NN" [PER: 'Sarmishtha/NNP']
[PER: 'Vrishaparvan/NNP'] "'s/POS daughter/NN" [PER: 'Sarmishtha/NNP']
[PER: 'Vyasa/NNP'] ',/, after/IN having/VBG secured/VBN the/DT assent/NN of/IN her/PRP$ daughter-in-law/NN ./.' [PER: 'Vyasa/NNP']
[PER: 'Tapati/NNP'] ',/, the/DT daughter/NN of/IN the/DT god/NN' [PER: 'Vivaswat/NNP']
[PER: 'Surya/NNP'] 'regarded/VBD him/PRP as/IN the/DT fit/JJ husband/NN for/IN his/PRP$ daughter/NN ,/,' [PER: 'Tapati/NNP']
[PER: 'Tapana/NNP'] ',/, made/VBD over/IN his/PRP$ daughter/NN ,/,' [PER: 'Tapati/NNP']
[PER: 'Yudhishthira/NNP'] 'said/VBD ,/, ``/`` The/DT daughter/NN of/IN king/VBG' [PER: 'Yajnasena/NNP']
[PER: 'Vasudeva/NNP'] "'s/POS daughter/NN and/CC" [PER: 'Vasudeva/NNP']
```

Some relations drawn from above result -

- ***Sarmishtha*** is the daughter of ***Vrishaparvan***.
- ***Tapati*** is the daughter of ***Vivaswat***.

Brother :-



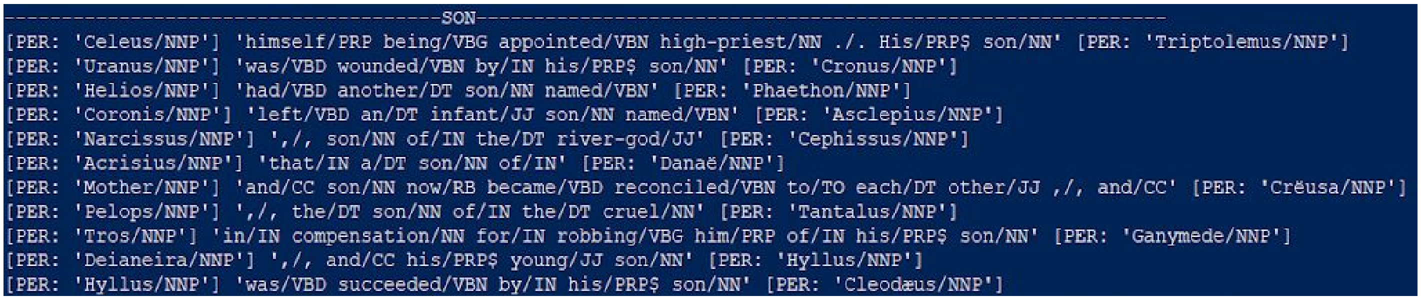
```
-----------------------------------BROTHER-------------------------------------------------------------
[PER: 'Hidimba/NNP'] ',/, and/CC the/DT slaying/NN of/IN her/PRP$ brother/NN' [PER: 'Hidimba/NNP']
[PER: 'Arjuna/NNP'] ',/, incited/VBD thereto/NN by/IN her/PRP$ brother/NN' [PER: 'Krishna/NNP']
[PER: 'Ambalika/NNP'] 'on/IN his/PRP$ younger/JJR brother/NN Vichitravirya/NNP ./. And/CC though/IN' [PER: 'Vichitravirya/NNP']
[PER: 'Utathya/NNP'] "'s/POS younger/JJR brother/NN" [PER: 'Vrihaspati/NNP']
[PER: 'Yudhishthira/NN'] 'and/CC his/PRP$ younger/JJR brother/NN' [PER: 'Arjuna/NNP']
```

Some relations drawn from above result -

- ***Arjuna*** and ***Krishna*** are brothers.
- ***Vichitravirya*** is the younger brother of ***Ambalika***.
- ***Vrihaspati*** is the younger brother of ***Utathya***.
- ***Arjuna*** is the younger brother of ***Yudhishthira***.

Book-2

Son :-

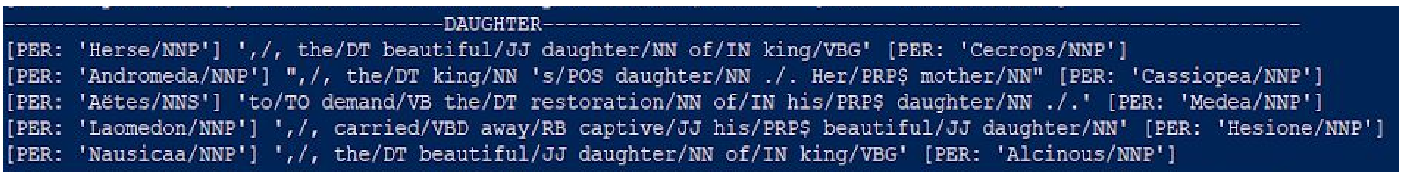


```
----------------------------------------SON----------------------------------------------------------------
[PER: 'Celeus/NNP'] 'himself/PRP being/VBG appointed/VBN high-priest/NN ./. His/PRP$ son/NN' [PER: 'Triptolemus/NNP']
[PER: 'Uranus/NNP'] 'was/VBD wounded/VBN by/IN his/PRP$ son/NN' [PER: 'Cronus/NNP']
[PER: 'Helios/NNP'] 'had/VBD another/DT son/NN named/VBN' [PER: 'Phaethon/NNP']
[PER: 'Coronis/NNP'] 'left/VBD an/DT infant/JJ son/NN named/VBN' [PER: 'Asclepius/NNP']
[PER: 'Narcissus/NNP'] ',/, son/NN of/IN the/DT river-god/JJ' [PER: 'Cephissus/NNP']
[PER: 'Acrisius/NNP'] 'that/IN a/DT son/NN of/IN' [PER: 'Danaë/NNP']
[PER: 'Mother/NNP'] 'and/CC son/NN now/RB became/VBD reconciled/VBN to/TO each/DT other/JJ ,/, and/CC' [PER: 'Crëusa/NNP']
[PER: 'Pelops/NNP'] ',/, the/DT son/NN of/IN the/DT cruel/NN' [PER: 'Tantalus/NNP']
[PER: 'Tros/NNP'] 'in/IN compensation/NN for/IN robbing/VBG him/PRP of/IN his/PRP$ son/NN' [PER: 'Ganymede/NNP']
[PER: 'Deianeira/NNP'] ',/, and/CC his/PRP$ young/JJ son/NN' [PER: 'Hyllus/NNP']
[PER: 'Hyllus/NNP'] 'was/VBD succeeded/VBN by/IN his/PRP$ son/NN' [PER: 'Cleodæus/NNP']
```

Some relations drawn from above result -

- ***Triptolemus*** is the son of ***Celeus***.
- ***Cronus*** is the son of ***Uranus***.
- ***Phaethon*** is the son of ***Helios***.
- ***Pelops*** is the son of ***Tantalus***.

Daughter :-

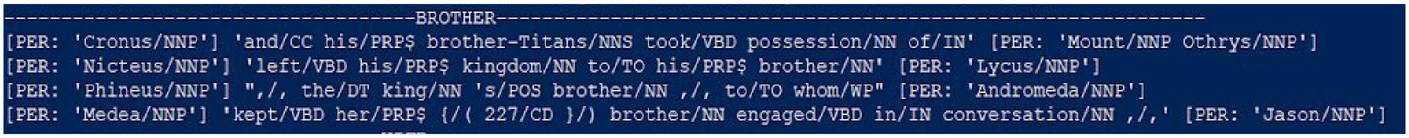


```
-----------------------------------DAUGHTER------------------------------------------------------------
[PER: 'Herse/NNP'] ',/, the/DT beautiful/JJ daughter/NN of/IN king/VBG' [PER: 'Cecrops/NNP']
[PER: 'Andromeda/NNP'] ",/, the/DT king/NN 's/POS daughter/NN ./. Her/PRP$ mother/NN" [PER: 'Cassiopea/NNP']
[PER: 'Aëtes/NNS'] 'to/TO demand/VB the/DT restoration/NN of/IN his/PRP$ daughter/NN ./.' [PER: 'Medea/NNP']
[PER: 'Laomedon/NNP'] ',/, carried/VBD away/RB captive/JJ his/PRP$ beautiful/JJ daughter/NN' [PER: 'Hesione/NNP']
[PER: 'Nausicaa/NNP'] ',/, the/DT beautiful/JJ daughter/NN of/IN king/VBG' [PER: 'Alcinous/NNP']
```

Some relations drawn from above result -

- ***Herse*** is the daughter of ***Cecrops***.
- ***Andromeda*** is the daughter of the king.
- ***Medea*** is the daughter of ***Aetes***.
- ***Hesione*** is the daughter of ***Laomedon***.

Brother :-



```
-----------------------------------BROTHER--------------------------------------------------------------
[PER: 'Cronus/NNP'] 'and/CC his/PRP$ brother-Titans/NNS took/VBD possession/NN of/IN' [PER: 'Mount/NNP Othrys/NNP']
[PER: 'Nicteus/NNP'] 'left/VBD his/PRP$ kingdom/NN to/TO his/PRP$ brother/NN' [PER: 'Lycus/NNP']
[PER: 'Phineus/NNP'] ",/, the/DT king/NN 's/POS brother/NN ,/, to/TO whom/WP" [PER: 'Andromeda/NNP']
[PER: 'Medea/NNP'] 'kept/VBD her/PRP$ {/( 227/CD }/) brother/NN engaged/VBD in/IN conversation/NN ,/,' [PER: 'Jason/NNP']
```

Some relations drawn from above result -

- ***Lycus*** and ***Nicteus*** are brothers.
- ***Phineus*** is the king's brother.

Inferences :-

- We have found a lot of entries of each relation for each book.
- Majority of them are relevant entries (true positives) like *"Pramati[PER] had a son named Ruru[PER]"* (6th entry in Son's for book 1) as clear relation can be observed between Pramati and Ruru.
- There are a some False positives, for example: *"Purochana[PER] was sleeping, then the son of Pandu[PER]"* As clearly observed there is no relation between *Purochana* and *Pandu*. This is due to the specified neighbourhood of our regular expression, as it is not taking the complete reference into account but only the surrounding words.
- In addition, we got some redundant entries as observed in "*Daughter*" relation of Book 1 like *"Vrishaparvan's daughter Sarmishtha"* is repeated 5 times since our data have recurring sentences.

- Also we got restricted relations, i.e, if there is any sentence like "*Ram had 3 brothers Laxman, Bharat, and Shatrughan*", then our algorithm will only return "*Ram had 3 brothers Laxman"* according to our regular expression used as it stops when it finds first PER entity.

References :-

- NLP class lectures
- https://www.nltk.org/api/nltk.sem.html?highlight=extract_rels#nltk.sem.relextract.extract_rels
- https://www.nltk.org/api/nltk.chunk.html?highlight=ne_chunk#nltk.chunk.ne_chunk
- https://www.nltk.org/book/ch05.html
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html