NLP PROJECT

ROUND 1 REPORT

TEAM NAME : Nlp_Learners

MOHIT JAIN (19UCS050)
AYUSH KABRA (19UCS113)
ARYAN VYAS (19UCS117)

Github repository: https://github.com/Aryan-Vyas/NlpLearners_NlpProject

Books used :

1. A town is drowning….by frederik pohl : T1
2. Nick Carter Stories….by Nick Carter : T2

TASKS
1. Import the text from two books, let's call it as T1 and T2.
2. Perform simple text pre-processing steps and tokenize the text T1 and T2.
3. Analyze the frequency distribution of tokens in T1 and T2 separately.
4. Create a Word Cloud of T1 and T2 using the token that you have got.
5. Remove the stopwords from T1 and T2 and then again create a word cloud.
6. Compare with word clouds before the removal of stopwords.
7. Evaluate the relationship between the word length and frequency for both T1 and T2.
8. Do PoS Tagging for both T1 and T2 using anyone of the four tagset studied in the class and Get the distribution of various tags

Python Libraries used in this project:

NLTK - Used for Tokenizing, Lemmatization and Removing Stopwords
Re - Used to remove URLs and Decontract Contractions in English Language
Wordcloud - Used to create WordClouds from Tokenized Data
Matplotlib - Used to Visualize our text data

DATA PREPROCESSING STEPS

1. Removing not needed text of Book
We will remove the documentation part of the text that is of no use to us.
2. Remove chapter name
We will remove chapter names as given in the task.
3. Convert all data to lowercase
We will convert all text data to lowercase.
4. Remove link
We will remove urls using regular expression
5. Remove Punctuations
6. Lemmatization
We do Lemmatization with the help of WordNetLemmatizer()function

Data Preparation

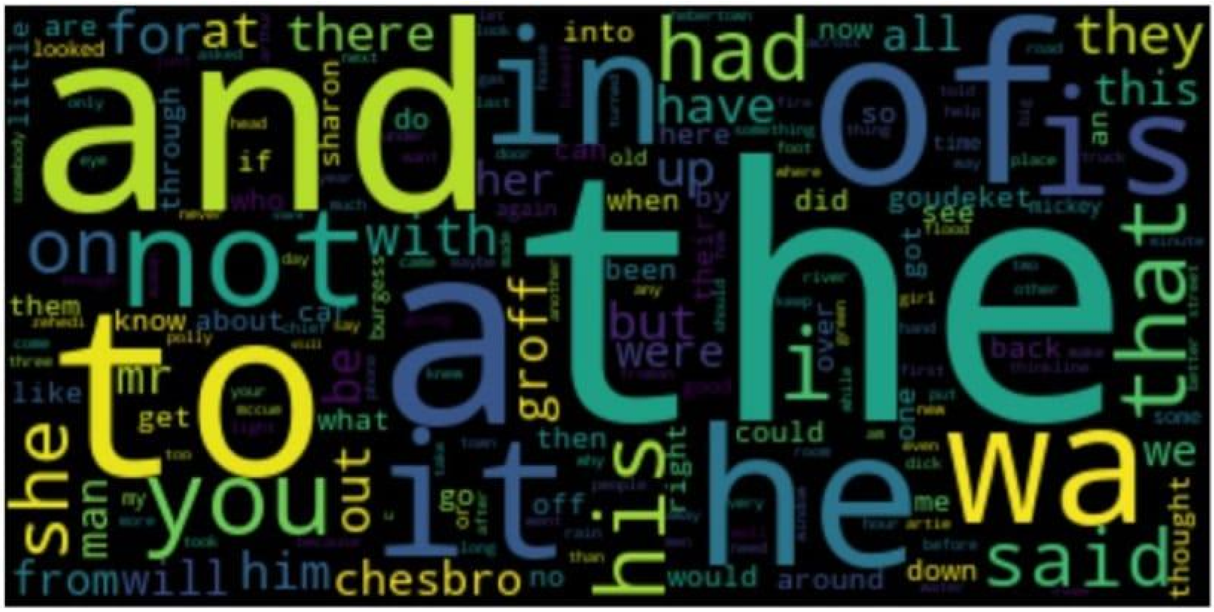
We apply all the functionalities we added above and Prepare our data for analysis

Problem Statements and Inferences

- Tokenization
- Analyze the frequency distribution of tokens in T1 and T2 separately
  we use python library pandas First we tokenize the given data, and then we plot a bargraph of top 10 most frequent words

  Generating a word cloud from T1 and T2
  For this we use python library wordcloud and its function WordCloud.

For T1


the
and
a
to
of
in
is
wa
he
it
you
not
that
his
had
on
said
they
there
at
for
with
have
this
her
up
but
were
i
chesbro
groff
she
him
from
will
out
be
mr
man
all
now
into
when
could
would
around
down
thought
we
me
then
what
get
know
about
them
like
did
goudeket
so
here
do
if
through
sharon
little
are
looked
been
see
back
off
go
one
right
no
by
car
burgess
my
their
over
got

For T2


the
and
to
of
in
a
he
wa
that
his
not
you
had
it
with
nick
have
i
is
for
at
this
said
which
him
on
be
were
could
up
her
been
then
one
chick
there
by
but
carter
floyd
what
they
from
she
or
so
out
would
an
me
them
when
who
more
no
all
are
will
grail
old
know
patsy
get
see
after
girl
time
well
down
other
put
waldmere
face
if
only
about
two
into
some
colonel
year
bannon
much
than
over
man
up
your
we

Inferences

- Words like ‘and’, ‘a’, ‘of’, ‘to’ and ‘the’ are the most frequently used words in T1
- Words like ‘and’, ‘the’, ‘he’, ‘to’’ are frequently used words in T2
- The words seen in wordcloud are called stop words

Generating new word clouds after removing stopwords

To remove stopwords, we useSTOPWORDS function in nltk.

For T1

For T2

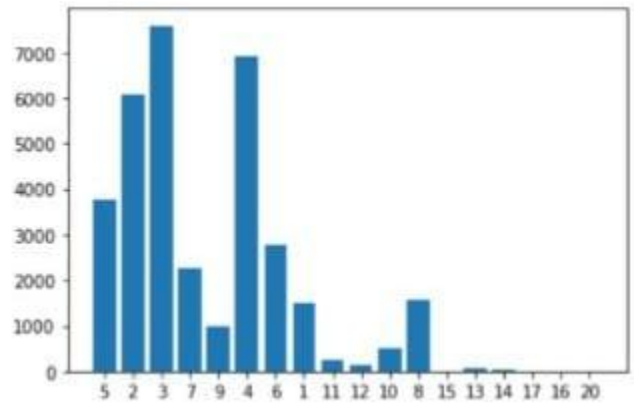

said
nick
one
carter
floyd
patsy
waldmere
must
much
sure
look
grail
year
may
make
two
three
old
car
plate
take
colonel
face
know
hand
see
put
eye
right
bannon
schilder
way
back
made
man
girl
crook
detective
new
sir
got
state
minute
men
woman
matter
away
minerva
though
nothing
still
think
first
found
river
gilbert
toward
flynn
thought
want
better
door
part
mind
last
thing
office
turned
mean
quickly
light
say
place
never
major
note
room
took
vera
melting
might
far
day
lost
appeared
near
half
head
going
im
even
dead
sent
left
find
ball
ye
nine
suspicion
stuart
ive
kind
small
several
give
night
house
wanted
known
train
chief
asked
fact
done
feet
let
dark
quite
closed
rather
york
upon
work
show
another
heard
name
later
taken
continued
wall
ago
desk
looking
nearly
city
order
bit
doubt
sight
john
given
questioned
person
adjutant
u
grand
0
200
400
600
800
1000

Inferences

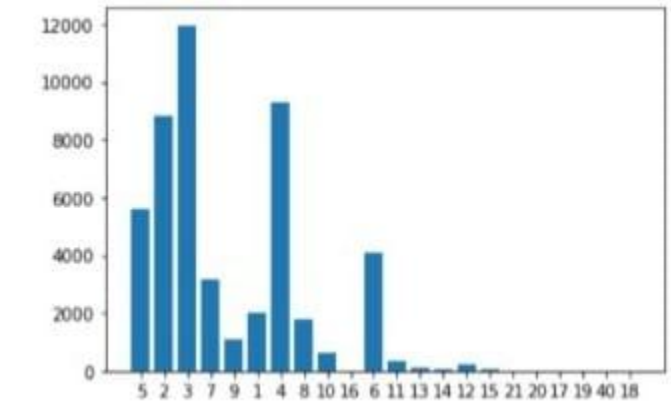
- Now the stopwords are removed
- New words like ‘sharon’, ‘groff’ in T1 and ‘nick’, ‘carter’, ‘floyd’, etc. can be seen in wordcloud which indicates the name of characters and other important stuffs around whom the book revolves.
- Therefore, after removing stopwords we get more meaning and understanding of the book.

Evaluate the relationship between the word length and frequency for both T1 and T2.

For T1


7000
6000
5000
4000
3000
2000
1000
0
5 2 3 7 9 4 6 1 11 12 10 8 15 13 14 17 16 20

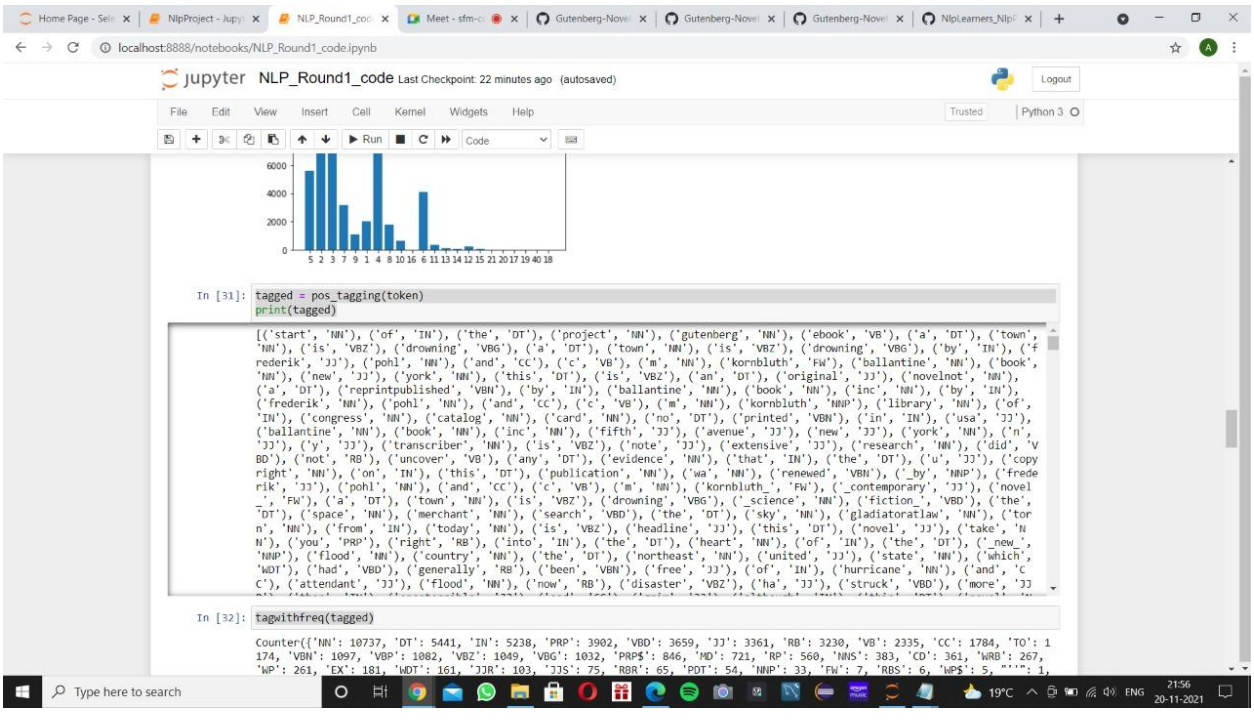
For T2


12000
10000
8000
6000
4000
2000
0
5 2 3 7 9 1 4 8 10 16 6 11 13 14 12 15 21 20 17 19 40 18

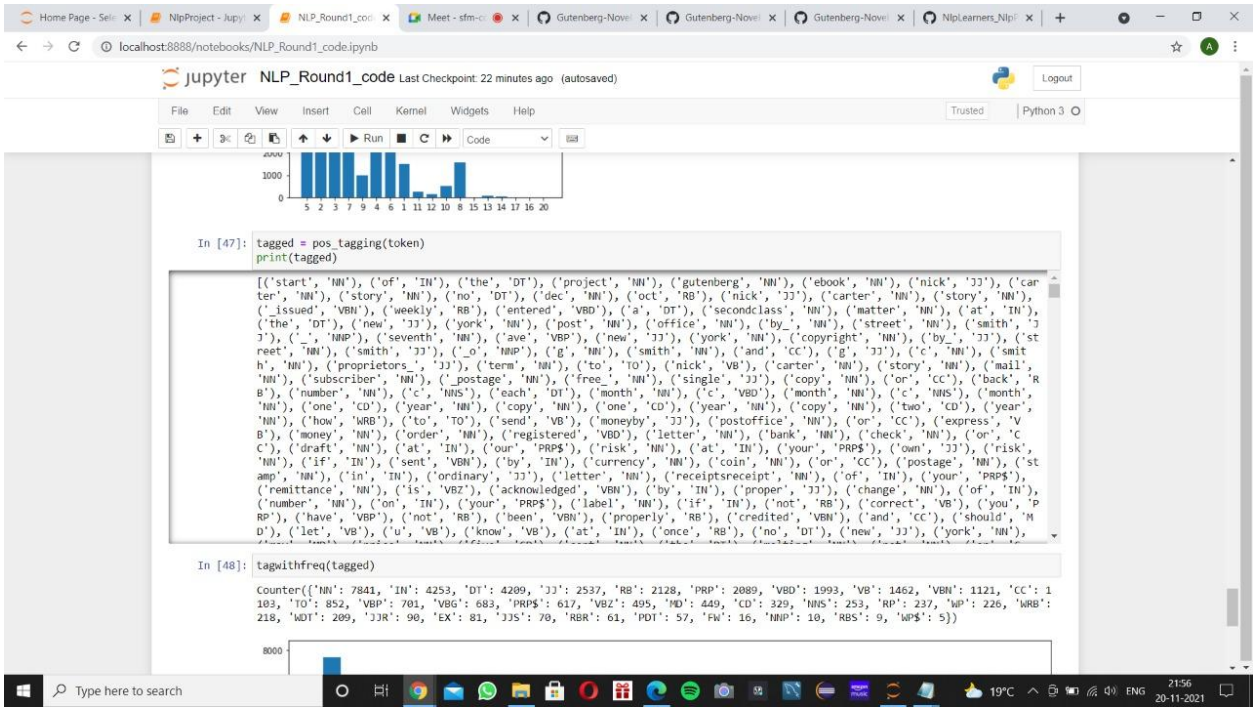
Performing POS Tagging

We will now perform the POS Tagging on T1 and T2 using inbuilt functions of nltk namely post_tag() which uses Penn Treebank tag set to perform POS tagging.

For T1



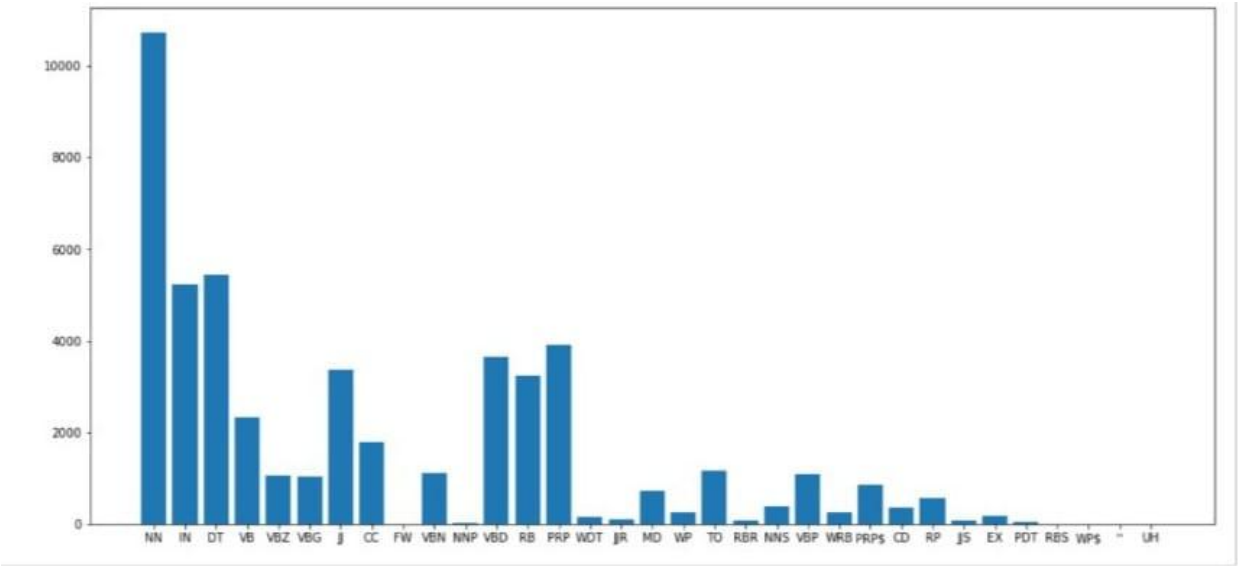
For T2


jupyter NLP_Round1_code Last Checkpoint: 22 minutes ago (autosaved)
In [47]: tagged = pos_tagging(token)
print(tagged)
[('start', 'NN'), ('of', 'IN'), ('the', 'DT'), ('project', 'NN'), ('gutenberg', 'NN'), ('ebook', 'NN'), ('nick', 'JJ'), ('carter', 'NN'), ('story', 'NN'), ('_issued', 'VBN'), ('weekly', 'RB'), ('entered', 'VBD'), ('a', 'DT'), ('secondclass', 'NN'), ('matter', 'NN'), ('at', 'IN'), ('the', 'DT'), ('new', 'JJ'), ('york', 'NN'), ('post', 'NN'), ('office', 'NN'), ('by_', 'NN'), ('street', 'NN'), ('smith', 'JJ'), ('_', 'NNP'), ('seventh', 'NN'), ('ave', 'VBP'), ('new', 'JJ'), ('york', 'NN'), ('copyright', 'NN'), ('by_', 'JJ'), ('street', 'NN'), ('smith', 'JJ'), ('_o', 'NNP'), ('g', 'NN'), ('smith', 'NN'), ('and', 'CC'), ('g', 'JJ'), ('c', 'NN'), ('smith', 'NN'), ('proprietors_', 'JJ'), ('term', 'NN'), ('to', 'TO'), ('nick', 'VB'), ('carter', 'NN'), ('story', 'NN'), ('mail', 'NN'), ('subscriber', 'NN'), ('_postage', 'NN'), ('free_', 'NN'), ('single', 'JJ'), ('copy', 'NN'), ('or', 'CC'), ('back', 'RB'), ('number', 'NN'), ('c', 'NNS'), ('each', 'DT'), ('month', 'NN'), ('c', 'VBD'), ('month', 'NN'), ('c', 'NNS'), ('month', 'NN'), ('one', 'CD'), ('year', 'NN'), ('copy', 'NN'), ('one', 'CD'), ('year', 'NN'), ('copy', 'NN'), ('two', 'CD'), ('year', 'NN'), ('how', 'WRB'), ('to', 'TO'), ('send', 'VB'), ('moneyby', 'JJ'), ('postoffice', 'NN'), ('or', 'CC'), ('express', 'VB'), ('money', 'NN'), ('order', 'NN'), ('registered', 'VBD'), ('letter', 'NN'), ('bank', 'NN'), ('check', 'NN'), ('or', 'CC'), ('draft', 'NN'), ('at', 'IN'), ('our', 'PRP$'), ('risk', 'NN'), ('at', 'IN'), ('your', 'PRP$'), ('own', 'JJ'), ('risk', 'NN'), ('if', 'IN'), ('sent', 'VBN'), ('by', 'IN'), ('currency', 'NN'), ('coin', 'NN'), ('or', 'CC'), ('postage', 'NN'), ('stamp', 'NN'), ('in', 'IN'), ('ordinary', 'JJ'), ('letter', 'NN'), ('receiptsreceipt', 'NN'), ('of', 'IN'), ('your', 'PRP$'), ('remittance', 'NN'), ('is', 'VBZ'), ('acknowledged', 'VBN'), ('by', 'IN'), ('proper', 'JJ'), ('change', 'NN'), ('of', 'IN'), ('number', 'NN'), ('on', 'IN'), ('your', 'PRP$'), ('label', 'NN'), ('if', 'IN'), ('not', 'RB'), ('correct', 'VB'), ('you', 'PRP'), ('have', 'VBP'), ('not', 'RB'), ('been', 'VBN'), ('properly', 'RB'), ('credited', 'VBN'), ('and', 'CC'), ('should', 'MD'), ('let', 'VB'), ('u', 'VB'), ('know', 'VB'), ('at', 'IN'), ('once', 'RB'), ('no', 'DT'), ('new', 'JJ'), ('york', 'NN'),
In [48]: tagwithfreq(tagged)
Counter({'NN': 7841, 'IN': 4253, 'DT': 4209, 'JJ': 2537, 'RB': 2128, 'PRP': 2089, 'VBD': 1993, 'VB': 1462, 'VBN': 1121, 'CC': 1103, 'TO': 852, 'VBP': 701, 'VBG': 683, 'PRP$': 617, 'VBZ': 495, 'MD': 449, 'CD': 329, 'NNS': 253, 'RP': 237, 'WP': 226, 'WRB': 218, 'WDT': 209, 'JJR': 90, 'EX': 81, 'JJS': 70, 'RBR': 61, 'PDT': 57, 'FW': 16, 'NNP': 10, 'RBS': 9, 'WP$': 5})
Type here to search
19°C
ENG
21:56
20-11-2021

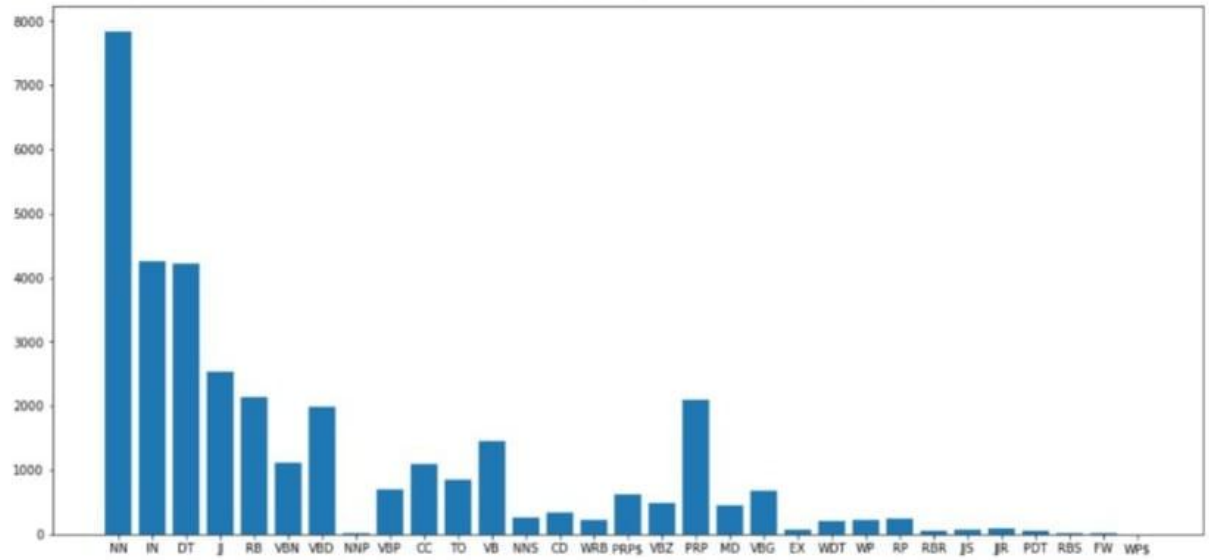
Frequency Distribution of Tags

Now, we plot the frequency distribution of tags after POS tagging on T1 and T2. For this we take the help of Counter() function from collections python library and FreqDist() function from nltk python library.

FOR T1



FOR T2



Inferences

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