NLP ROUND 1 REPORT:

TEAM NAME:

STARKS

TEAM MEMBERS:

KUBER VARSHNEY 19UCS194

SAATVIK SINGH 19UCS133

SAMARPIT SACHAN 19UCS036

Code Link:

https://github.com/kubercodes/NLP-Project/blob/main/NLP_Project.ipynb

Data Description:

Books used:

BOOK 1: Williwaw: A Novel by Gore Vidal.

BOOK 2: Hans Andersen's Fairy Tales by H. C. Andersen.

PROBLEM STATEMENT:

1. Import the text, let us call it as T1 and T2 (books that you have downloaded)

```
with open("/content/book1.txt") as f1:
  T1 = f1.read()
```

```
with open("/content/book2.txt") as f2:
  T2 = f2.read()
```

2. Perform simple text pre-processing steps and tokenize the text T1 and T2 — you may have to do the removal of running section / chapter names and so on.

- Tokenization

Here we are tokenizing the two books where we are splitting it into individual terms. Following are the codes snippets for tokenizing the books.

```
# Tokenizing T1
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
text = open('book1.txt',mode='r',encoding='utf-8').read()
T1_token = token = word_tokenize(text)
T1_token
```

```
# Tokenizing T2
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
text = open('book2.txt',mode='r',encoding='utf-8').read()
T2_token = token = word_tokenize(text)
T2_token
```

Snippets of the output:

Tokenisation of Book 1:

First 10 tokens:

```
'\ufeffThe',
'Project',
'Gutenberg',
'eBook',
'of',
'Williwaw',
',',
'by',
'Gore',
'Vidal',
```

Last 10 tokens:

```
',',
'll',
'go',
'below',
'now',
',',
'”',
'said',
'Evans',
'.',
```

Tokenisation of Book 2:

First 10 tokens:

```
'\ufeffThe',
'Project',
'Gutenberg',
'eBook',
'of',
'Hans',
'Andersen',
"'s",
'Fairy',
'Tales',
```

Last 10 tokens:

```
'said',
'the',
'king',
'253',
'Their',
'slippers',
'flew',
'about',
'their',
'ears',
```

- Remove headings and unwanted lines:

Lines starting with a new line or special characters are being removed in this piece of code.

For example:

```
_A Novel_

                          *       *       *       *       *


_Chapter One_
```

Code snippet:

```
# Here we are removing the headings and new lines.
# Lines starting with _ , * , [ and further more are being removed
for line in f.readlines():
  line = line.replace('\n',' ')         # removing blank lines
  #line = line.replace('',' ')
  l_list = []
  l_list = line.split(" ")
  if not (line.startswith('_') or line.startswith('       *') or line.startswith('[')
   or line.startswith('*') or line.startswith('ii') or l_list[0] == '\n' or l_list[0] == 'i\n'):
    f1.write(line)
f.close()
f1.close()
```

- Remove the punctuation marks:

Now we clean the different types of punctuation marks present in the text so that we are only left with letters.

Code snippet:

```
# Removing the punctuation marks and
# Removing ’ , ” , “ , ‘
for s in f.readlines():
  for char in string.punctuation:
    s = s.replace(char, ' ')
    s = s.replace("’","")
    s = s.replace("”","")
    s = s.replace("“","")
    s = s.replace("‘","")
```

- Remove chapter headings:

Chapter headings in capital letters are removed and cleaned here. It is one of the ways to get rid of the running sections in the book.

For example:

```
THE MARSH KING'S DAUGHTER
```

Code snippet:

```
def onlyUpper(word):
    for c in word:
        if not c.isupper():
            return False
    return True
```

```
# removing chapter headings as they are in capital letters
for w in words:
  if not onlyUpper(w):
    good_words.append(w)
```

- Dealing with Uppercase alphabets.

Uppercase and lowercase have different meanings so we need to change the uppercase alphabet into lowercase alphabets like The -> the or from It -> it and such many other examples.

```
# converting capital letter alphabet from words to lower case like The -> the
  result = ""
  for w in good_words:
    w = w.lower()
    result = result + w + " "

  f1.write(result)
```

3. Analyse the frequency distribution of tokens in T1 and T2 separately.

Code snippet from book 1:

```
from collections import Counter
with open("/content/book1_1.txt") as f1:
  T1 = f1.read()
T1_list = T1.split(" ")
counts = Counter(T1_list)
print(counts)
```

Code snippet from book 2:

```
from collections import Counter
with open("/content/book2_1.txt") as f2:
  T2 = f2.read()
T2_list = T2.split(" ")
counts = Counter(T2_list)
print(counts)
```

T1 frequency counts:

```
Counter({'the': 3977, 'he': 1606, 'to': 1320, 'and': 1204, 'was': 1058, 'of': 1023
```

T2 frequency counts:

```
Counter({'the': 5216, 'and': 2787, 'to': 1575, 'of': 1512, 'a': 1291, 'her': 1095,
```

4. Create a Word Cloud of T1 and T2 using the token that you have got.

Word Clouds are visual representations of words that give greater prominence to words that appear more frequently.

Code snippet for book 1:

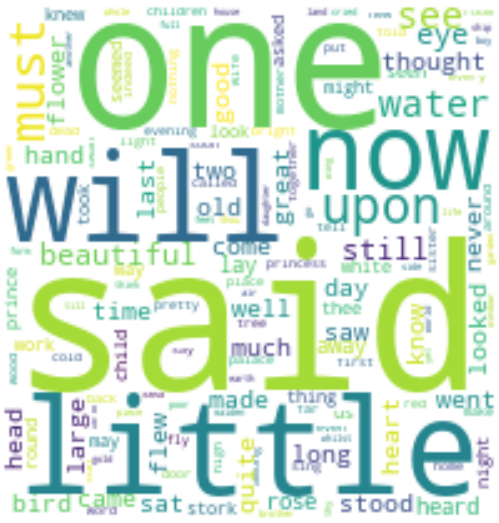
```
# Creating Word Cloud
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
import sys,os
text = open('book1_1.txt',mode='r',encoding='utf-8').read()
wc = WordCloud(
    background_color='white',
    height = 200,
    width=200
)
wc.generate(text)
wc.to_file('wordcloud_1.png')
```

Code snippet for book 2:

```
# Creating Word Cloud
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
import sys,os
text = open('book2_1.txt',mode='r',encoding='utf-8').read()
wc = WordCloud(
    background_color='white',
    height = 200,
    width=200
)
wc.generate(text)
wc.to_file('wordcloud_2.png')
```

T1 word cloud:

T2 word cloud:



5. Remove the stop words from T1 and T2 and then again create a word cloud.

Stopwords are the words in any language which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence.

We have used **NTLK** library of python to help us remove the stopwords from our text data.

```
# Importing stopwords from the nltk library
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
print(stopwords.words('english'))
```

```
# Removing stopwords
word_token = word_tokenize(text)
filter_word = []
for w in word_token:
  if w not in stop_word:
    f1.write(w)
    f1.write(" ")
f1.close()
```

Some examples of Stopwords:

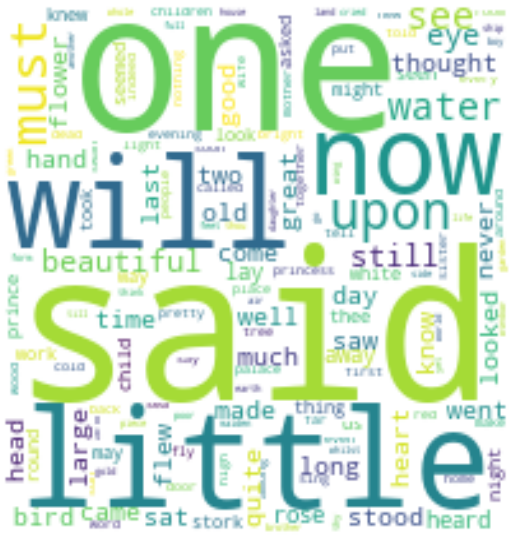
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'

'you', "you're", "you've", "you'll", "you'd", 'your', 'yours'
```

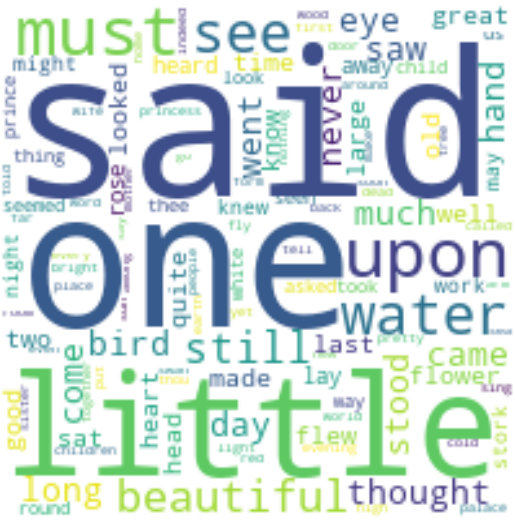
T1 word cloud before removing stop words:

T1 word cloud after removing stop words:

T2 word cloud before removing stop words:



T2 word cloud after removing stop words:



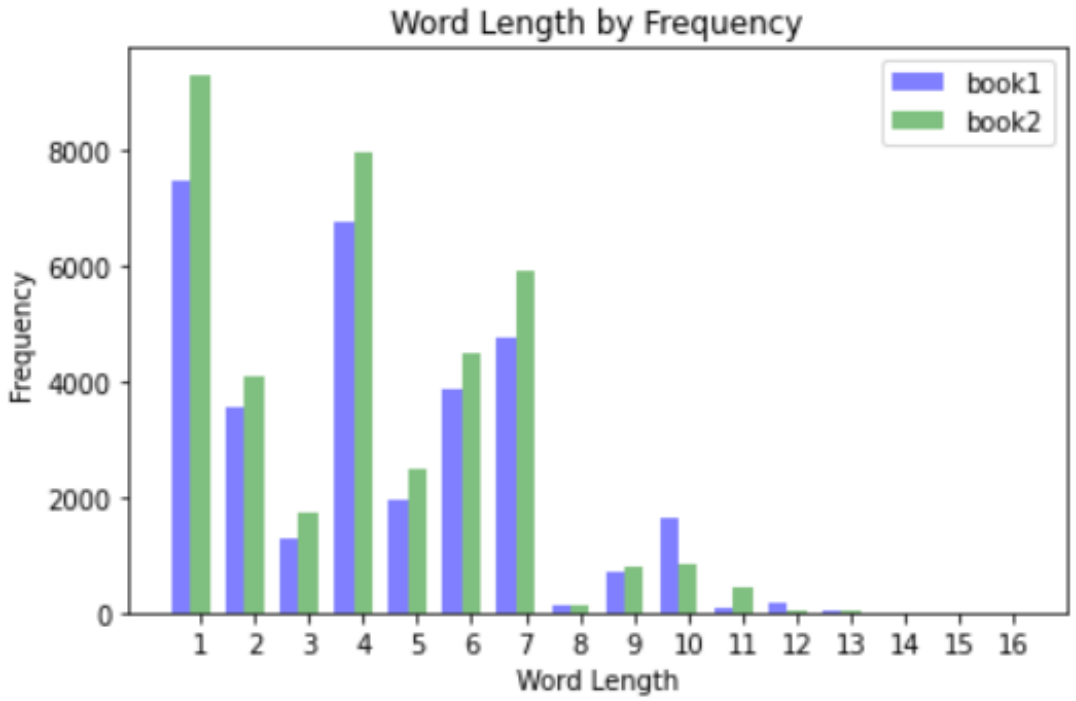
6. Evaluate the relationship between the word length and frequency for both T1 and T2.

Code Snippets:

```
text = open('book1_2.txt',mode='r',encoding='utf-8').read()
word_token = word_tokenize(text)
dic1 = {}
for char in word_token:
  #if(len(char) == 14):
     # print(char)
  if dic1.get(len(char),None) == None:
    dic1[len(char)] = 1
  else:
    dic1[len(char)] += 1
```

```
text = open('book2_2.txt',mode='r',encoding='utf-8').read()
word_token = word_tokenize(text)
dic2 = {}
for char in word_token:
  #if(len(char) == 16):
    #print(char)
  if dic2.get(len(char),None) == None:
    dic2[len(char)] = 1
  else:
    dic2[len(char)] += 1
```

Comparing word length and frequency of both books



7. Do PoS Tagging for both T1 and T2 using anyone of the four tag sets studied in the class and get the distribution of various tags.

Here we mark each word with their part of speech assigning them different tag sets.

```
# POS TAGGING in BOOK 1
from nltk.corpus import brown
brown_tagged_sents = brown.tagged_sents(categories='news')
brown_sents = brown.sents(categories='news')
unigram_tagger = nltk.UnigramTagger(brown_tagged_sents)
text = open('book1_2.txt',mode='r',encoding='utf-8').read()
word_token = word_tokenize(text)
unigram_tagger.tag(word_token)
```

```
# POS TAGGING in BOOK 2
from nltk.corpus import brown
brown_tagged_sents = brown.tagged_sents(categories='news')
brown_sents = brown.sents(categories='news')
unigram_tagger = nltk.UnigramTagger(brown_tagged_sents)
text = open('book2_2.txt',mode='r',encoding='utf-8').read()
word_token = word_tokenize(text)
unigram_tagger.tag(word_token)
```

BOOK 1:

Output snippets:

```
('Project', None),
('Gutenberg', None),
('eBook', None),
('Williwaw', None),
('Gore', 'NP'),
('Vidal', None),
('This', 'DT'),
('eBook', None),
('use', 'VB'),
('anyone', 'PN'),
('anywhere', 'RB'),
```

```
('Library', 'NN-TL'),
('By', 'IN'),
('1946', 'CD'),
('writing', 'VBG'),
('publisher', None),
('except', 'IN'),
('reviewer', None),
('wishes', None),
('quote', 'NN'),
('brief', 'JJ'),
('passages', 'NNS'),
```

BOOK 2:

Output snippets:

```
('Project', None),
('Gutenberg', None),
('eBook', None),
('Hans', None),
('Andersen', None),
('Fairy', 'NN-TL'),
('Tales', None),
('Hans', None),
('Andersen', None),
('This', 'DT'),
('eBook', None),
('use', 'VB'),
```

```
('entered', 'VBD'),
('large', 'JJ'),
('cold', 'JJ'),
('empty', None),
('hall', 'NN'),
('Tailpiece', None),
('The', 'AT'),
('elfin', None),
('kings', None),
('housekeeper', None),
('The', 'AT'),
('mer', None),
('king', 'NN'),
```