CAP6307

TEXT MINING ASSIGNMENT -1

Assignment 1

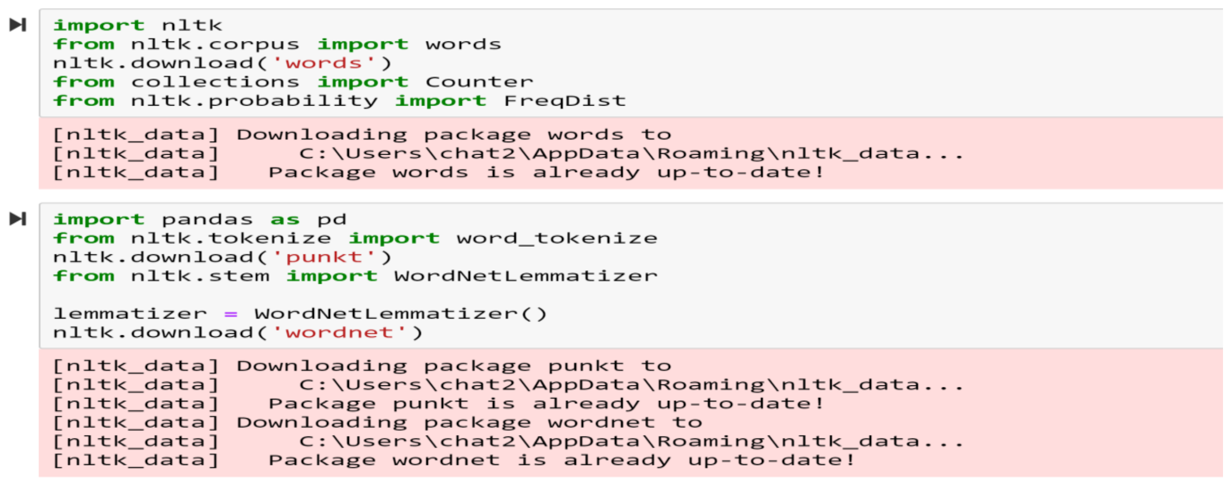
Introduction to NLP and Text Mining

Question 1:

Find Number of Tokens and Unique Tokens

Solution:

To find this, as prescribed I used NLTK library in python. Initially the file is read using pandas libray and using function word_tokenize() in NLTK library we can obtain the token. Unique tokens can be found by using np.unique() function in Numpy library. Total number of tokens is 255018 and number of unique tokens are 20754.

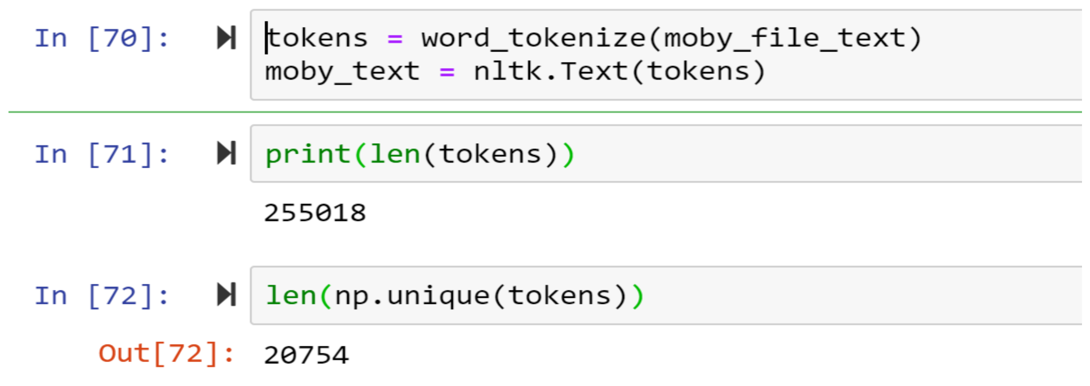


```
import nltk
from nltk.corpus import words
nltk.download('words')
from collections import Counter
from nltk.probability import FreqDist
```

```
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\chat2\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
```

```
import pandas as pd
from nltk.tokenize import word_tokenize
nltk.download('punkt')
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\chat2\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\chat2\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```



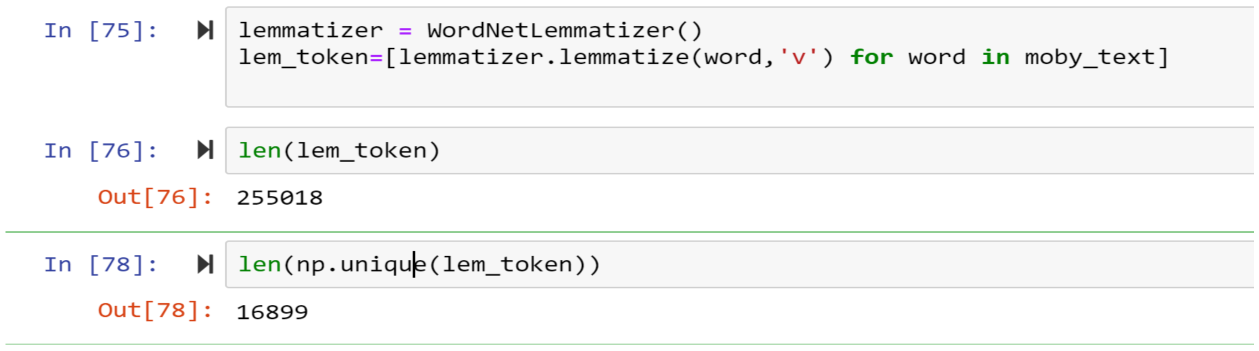
```
In [70]:  tokens = word_tokenize(moby_file_text)
          moby_text = nltk.Text(tokens)

In [71]:  print(len(tokens))

          255018

In [72]:  len(np.unique(tokens))

Out[72]:  20754
```

Question 2:

Calculate number of tokens after applying lemmatization on verbs.

Solution:

We import WordNetLemmatizer() function from NLTK package. This helps us in lemmatizing the verbs by adding ‘v’ as a parameter. We then find the length of tokens and unique tokens using the same above technique. We see that there are 255018 tokens and 16899 unique tokens after lemmatization.



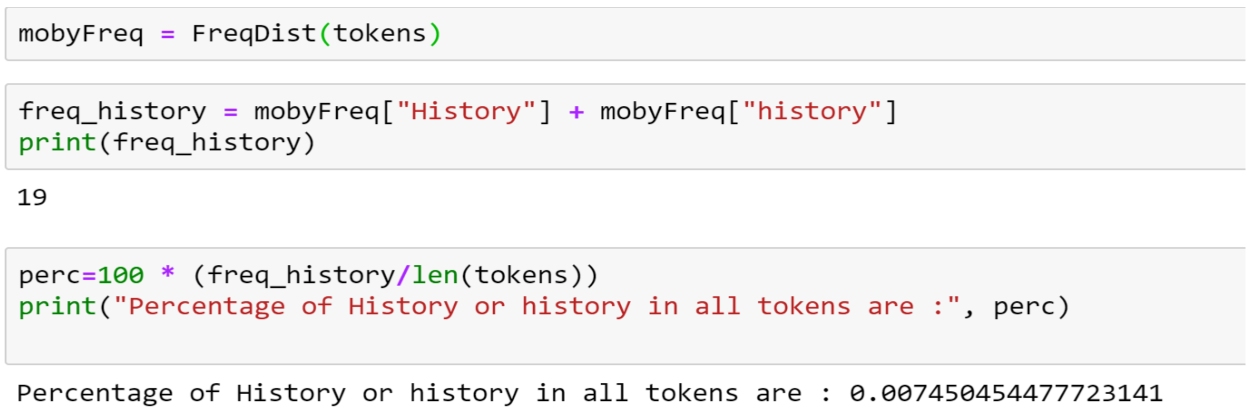
```
In [75]:  lemmatizer = WordNetLemmatizer()
           lem_token=[lemmatizer.lemmatize(word,'v') for word in moby_text]

In [76]:  len(lem_token)
Out[76]:  255018

In [78]:  len(np.unique(lem_token))
Out[78]:  16899
```

Question 3:

What percentage of History or history in all of tokens?

Solution:

We use FreqDist() function from NLTK.Probability library and obtain the frequency of words and then calculate total percentage by dividing with total tokens count. It has around 0.00745%



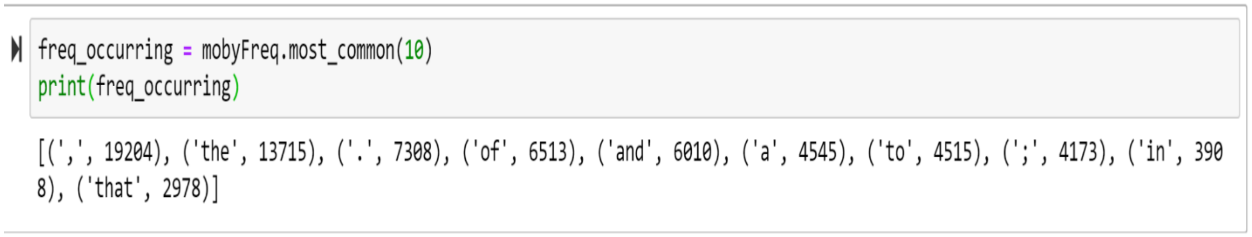
```
mobyFreq = FreqDist(tokens)

freq_history = mobyFreq["History"] + mobyFreq["history"]
print(freq_history)

19

perc=100 * (freq_history/len(tokens))
print("Percentage of History or history in all tokens are :", perc)

Percentage of History or history in all tokens are : 0.007450454477723141
```

Question 4:

Find Ten Most frequently occurring token

Solution:

We use most_common() function to find the top n most common occurring tokens in text.



```
freq_occurring = mobyFreq.most_common(10)
print(freq_occurring)

[(',', 19204), ('the', 13715), ('.', 7308), ('of', 6513), ('and', 6010), ('a', 4545), ('to', 4515), (';', 4173), ('in', 390
8), ('that', 2978)]
```

Spell Checker:

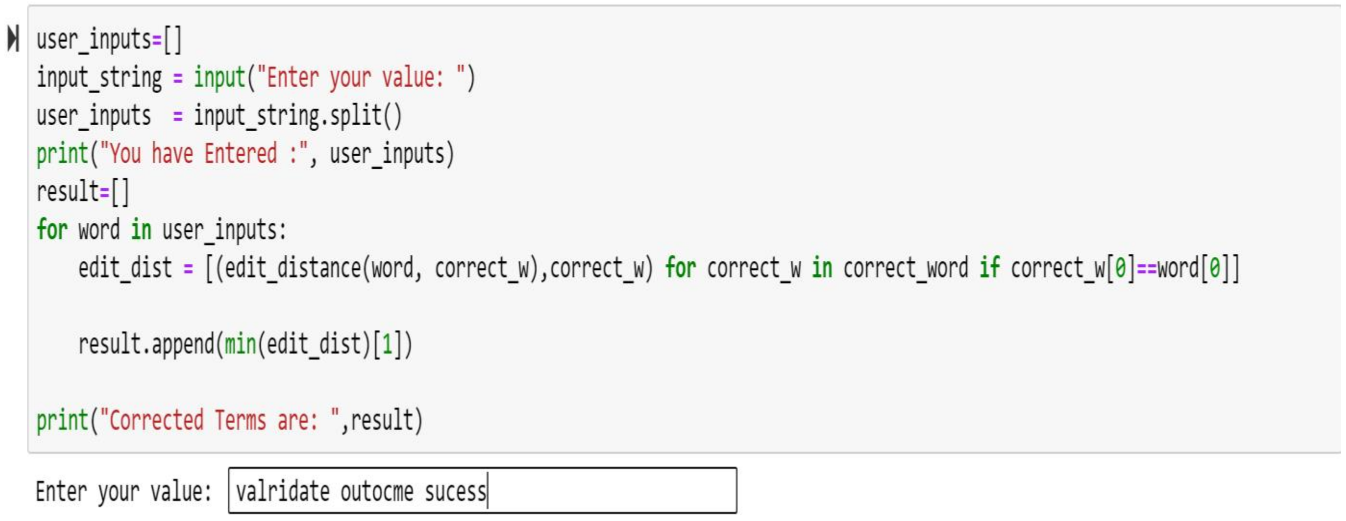
Provide Recommendation for the user.

Solution:

1) We initially download the word corpus from NLTK and import.
2) We also import edit_distance() from NLTK metrics library()
3) We then get the input from the user and split the number of words separately.
4) Then for each word entered we find the edit distance of them with the word in corpus.
5) We then get the word with minimum distance matching to the user input word and word in corpus and display to user.

Testing of Code:

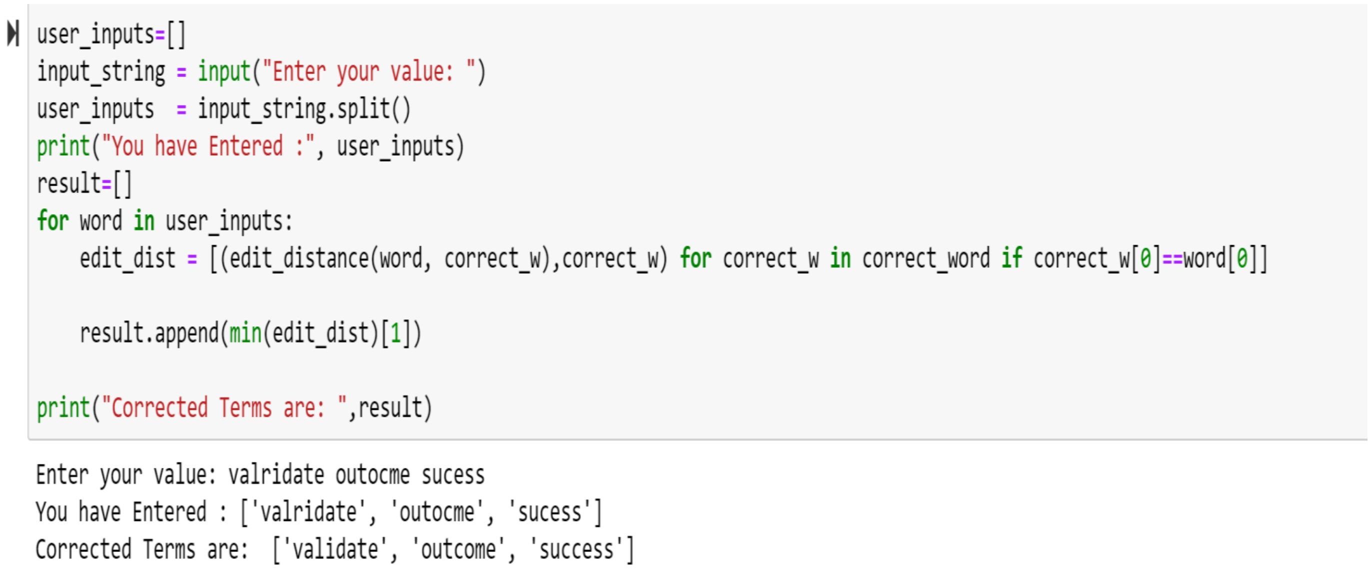
Code is tested by obtaining various inputs from user. Incorrect spelling like valridate, success and outome etc are provided as valid input.



```
user_inputs=[]
input_string = input("Enter your value: ")
user_inputs  = input_string.split()
print("You have Entered :", user_inputs)
result=[]
for word in user_inputs:
    edit_dist = [(edit_distance(word, correct_w),correct_w) for correct_w in correct_word if correct_w[0]==word[0]]

    result.append(min(edit_dist)[1])

print("Corrected Terms are: ",result)
```

```
Enter your value: valridate outocme sucess
```

We see that the corrected spellings are returned.



```
user_inputs=[]
input_string = input("Enter your value: ")
user_inputs  = input_string.split()
print("You have Entered :", user_inputs)
result=[]
for word in user_inputs:
    edit_dist = [(edit_distance(word, correct_w),correct_w) for correct_w in correct_word if correct_w[0]==word[0]]

    result.append(min(edit_dist)[1])

print("Corrected Terms are: ",result)
```

```
Enter your value: valridate outocme sucess
You have Entered : ['valridate', 'outocme', 'sucess']
Corrected Terms are:  ['validate', 'outcome', 'success']
```