Bhuvi Ghosh
60009210191

Experiment No 1

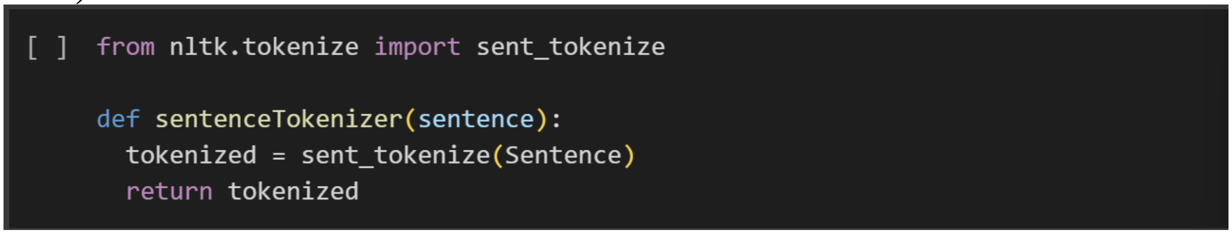
AIM: To study and implement Preprocessing of text (Tokenization, Filtration, ScriptValidation, Stop Word Removal, Stemming)

Lab Experiments to be Performed in This Session: -

Perform Following Preprocessing Techniques on the given corpus

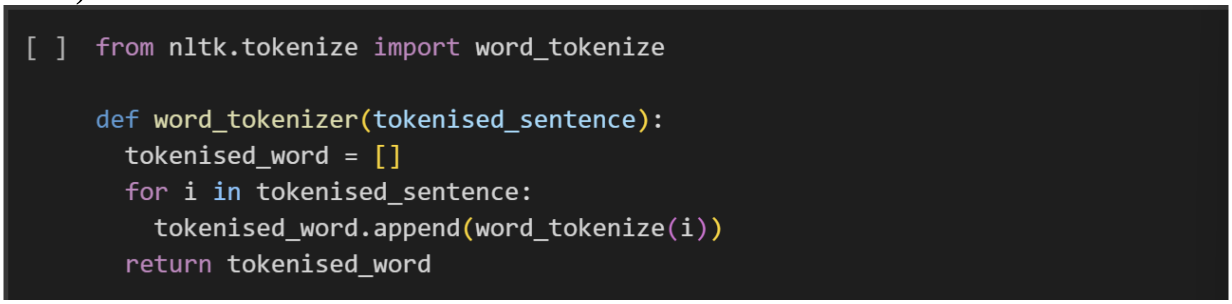
1. Tokenization,
2. Converting Text Lower Case
3. Remove Numbers
4. Converting Number to Words
5. Remove Punctuation
6. Remove Whitspaces
7. Remove StopWords
8. Count Word Frequency
9. Stemming (Porter Stemmer and Lancaster Stemmer)
10. Lemmatization

1.) Sentence Tokenizer



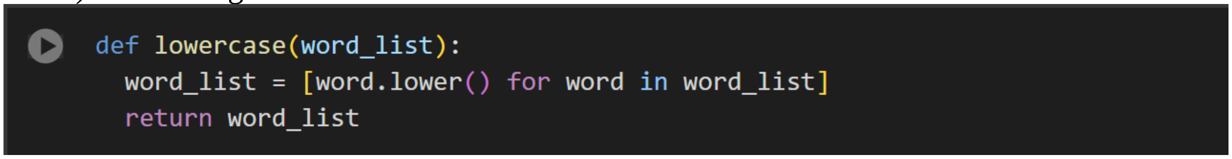
```
from nltk.tokenize import sent_tokenize

def sentenceTokenizer(sentence):
  tokenized = sent_tokenize(Sentence)
  return tokenized
```

2.) Word Tokenizer



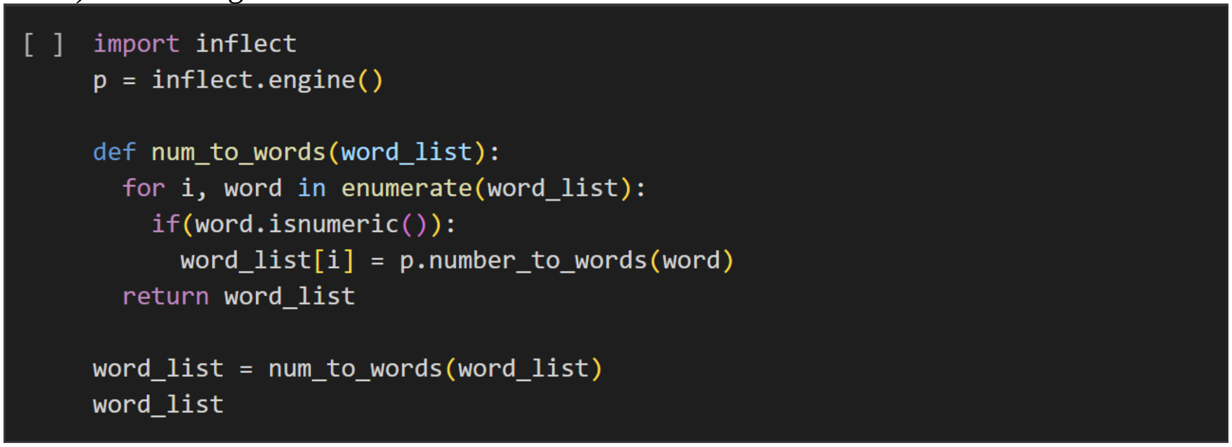
```
from nltk.tokenize import word_tokenize

def word_tokenizer(tokenised_sentence):
  tokenised_word = []
  for i in tokenised_sentence:
    tokenised_word.append(word_tokenize(i))
  return tokenised_word
```

3.) Converting Text Lower Case



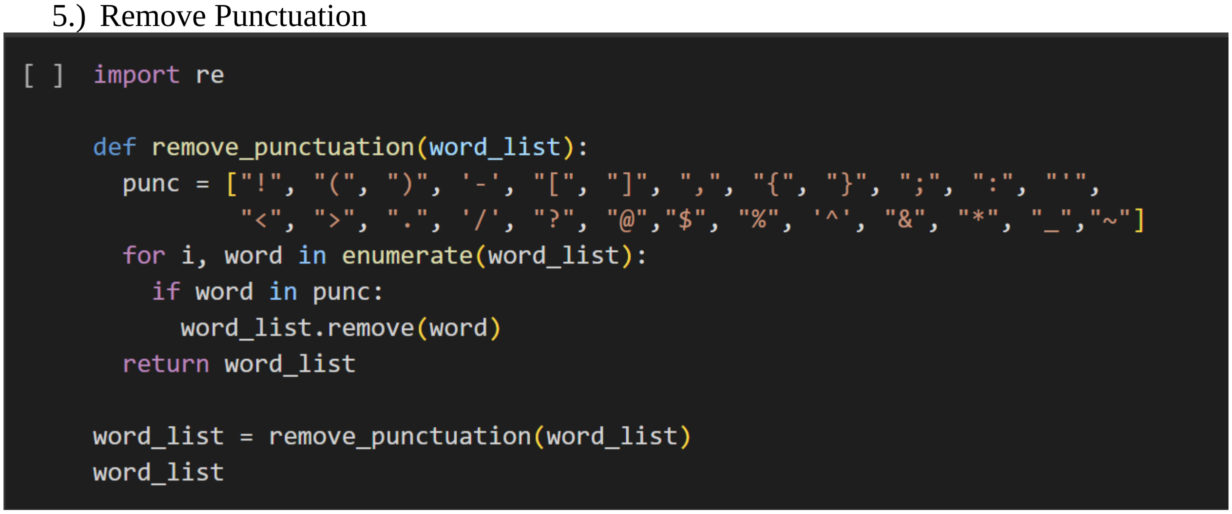
```
def lowercase(word_list):
  word_list = [word.lower() for word in word_list]
  return word_list
```

4.) Converting Number to Words



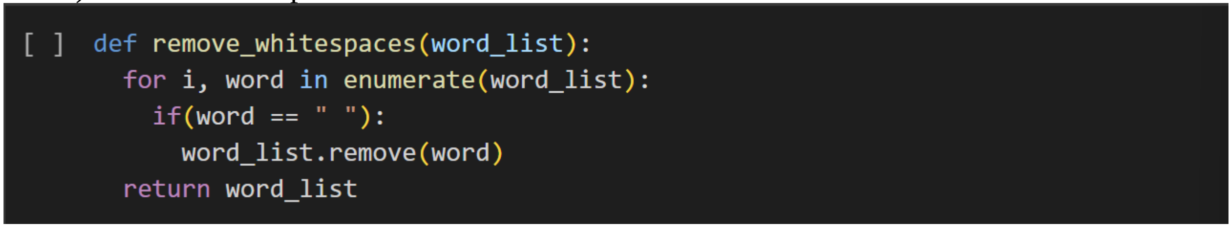
```
[ ]  import inflect
     p = inflect.engine()

     def num_to_words(word_list):
       for i, word in enumerate(word_list):
         if(word.isnumeric()):
           word_list[i] = p.number_to_words(word)
       return word_list

     word_list = num_to_words(word_list)
     word_list
```

5.) Remove Punctuation



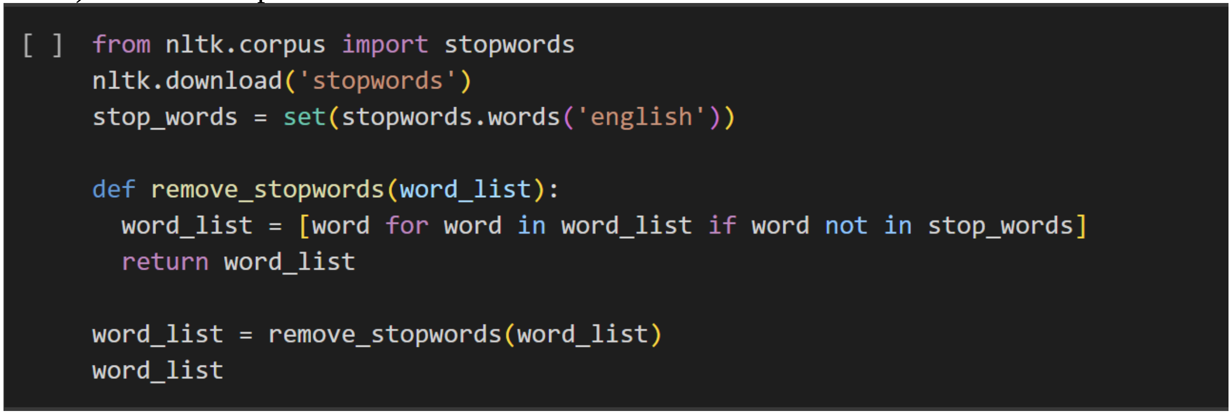
```
[ ]  import re

     def remove_punctuation(word_list):
       punc = ["!", "(", ")", '-', "[", "]", ",", "{", "}", ";", ":", "'",
               "<", ">", ".", '/', "?", "@","$", "%", '^', "&", "*", "_","~"]
       for i, word in enumerate(word_list):
         if word in punc:
           word_list.remove(word)
       return word_list

     word_list = remove_punctuation(word_list)
     word_list
```

6.) Remove Whitspaces



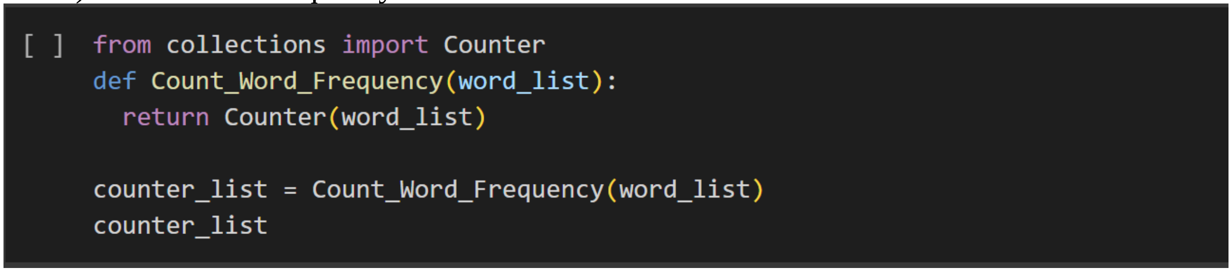
```
[ ]  def remove_whitespaces(word_list):
       for i, word in enumerate(word_list):
         if(word == " "):
           word_list.remove(word)
       return word_list
```

7.) Remove Stopwords



```
[ ]  from nltk.corpus import stopwords
     nltk.download('stopwords')
     stop_words = set(stopwords.words('english'))

     def remove_stopwords(word_list):
       word_list = [word for word in word_list if word not in stop_words]
       return word_list

     word_list = remove_stopwords(word_list)
     word_list
```

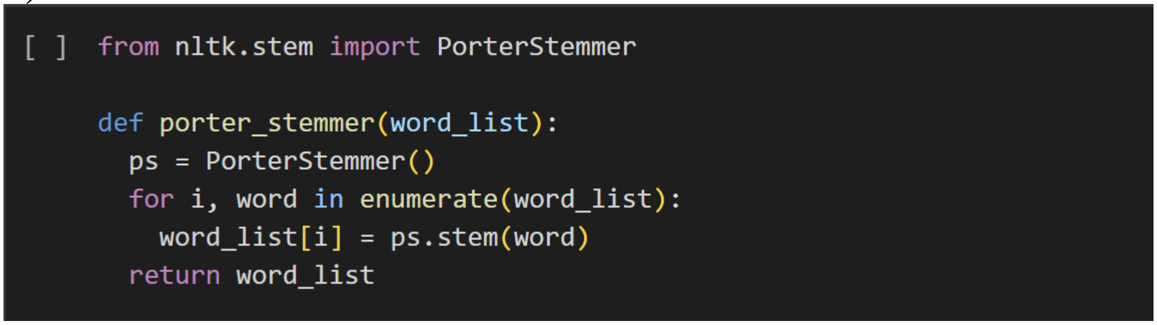
8.) Count word Frequency



```
from collections import Counter
def Count_Word_Frequency(word_list):
  return Counter(word_list)

counter_list = Count_Word_Frequency(word_list)
counter_list
```

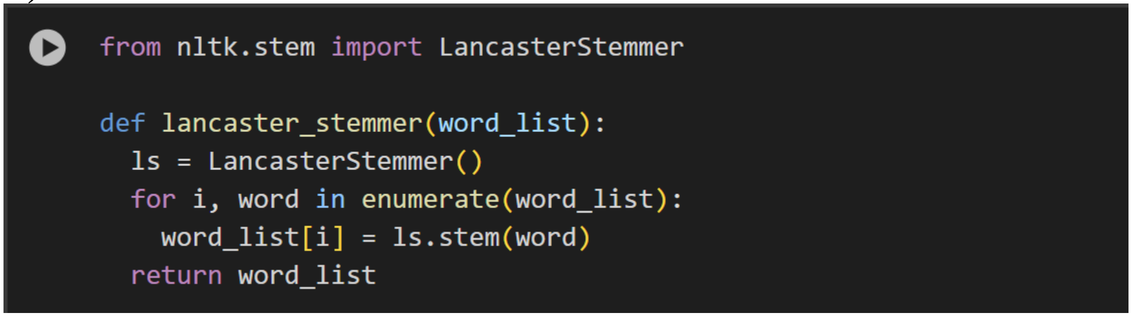
9.) Stemming

a.) Porter Stemmer



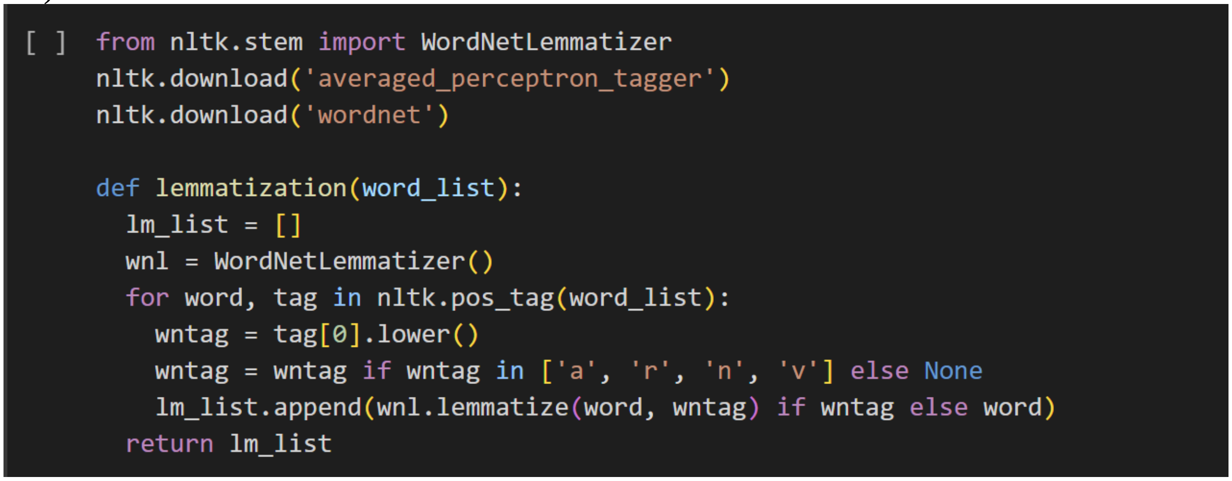
```
from nltk.stem import PorterStemmer

def porter_stemmer(word_list):
  ps = PorterStemmer()
  for i, word in enumerate(word_list):
    word_list[i] = ps.stem(word)
  return word_list
```

b.) Lancaster Stemmer



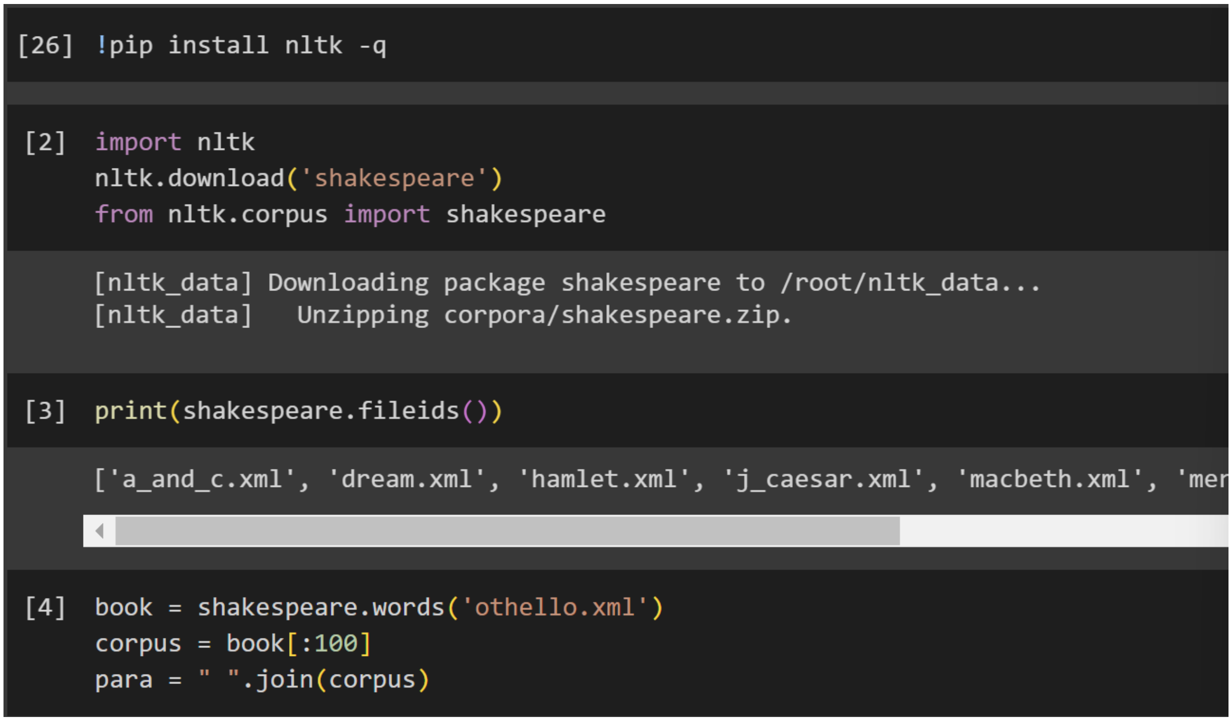
```
from nltk.stem import LancasterStemmer

def lancaster_stemmer(word_list):
  ls = LancasterStemmer()
  for i, word in enumerate(word_list):
    word_list[i] = ls.stem(word)
  return word_list
```

10.) Lematization



```
from nltk.stem import WordNetLemmatizer
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

def lemmatization(word_list):
  lm_list = []
  wnl = WordNetLemmatizer()
  for word, tag in nltk.pos_tag(word_list):
    wntag = tag[0].lower()
    wntag = wntag if wntag in ['a', 'r', 'n', 'v'] else None
    lm_list.append(wnl.lemmatize(word, wntag) if wntag else word)
  return lm_list
```

Performing All the steps on nltk.corpus => Shakespeare => Othello book dataset

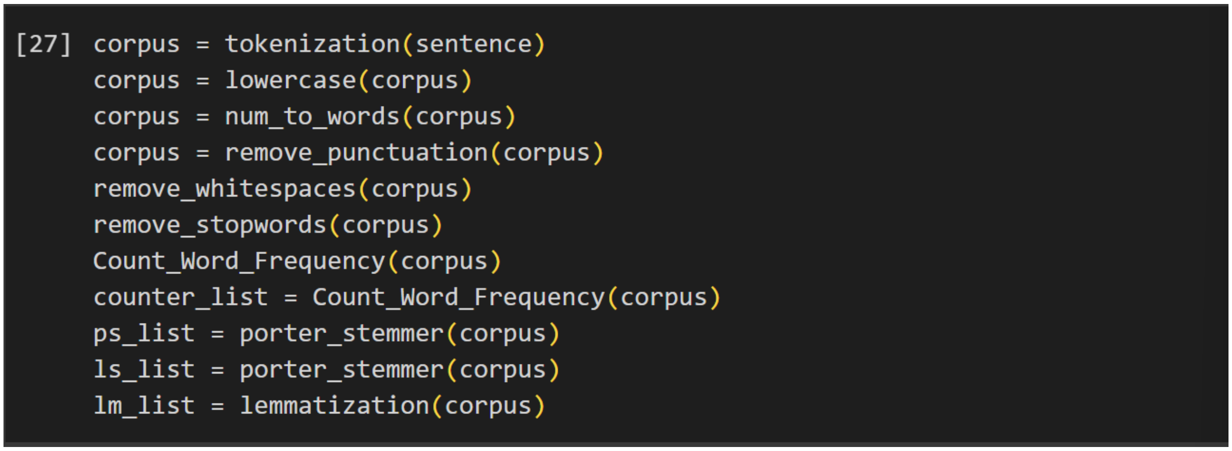


```
[26] !pip install nltk -q
```

```
[2]  import nltk
     nltk.download('shakespeare')
     from nltk.corpus import shakespeare

     [nltk_data] Downloading package shakespeare to /root/nltk_data...
     [nltk_data]   Unzipping corpora/shakespeare.zip.
```

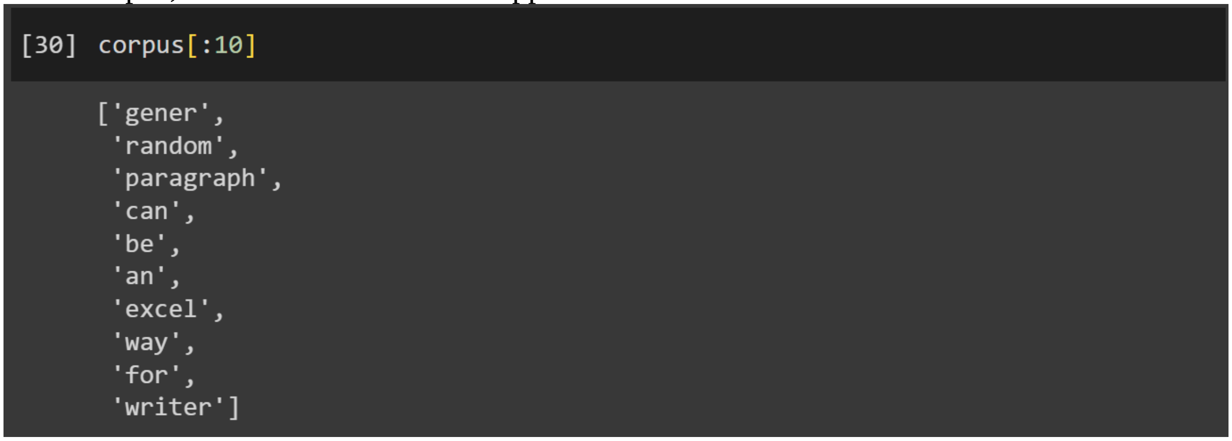
```
[3]  print(shakespeare.fileids())

     ['a_and_c.xml', 'dream.xml', 'hamlet.xml', 'j_caesar.xml', 'macbeth.xml', 'mer
```

```
[4]  book = shakespeare.words('othello.xml')
     corpus = book[:100]
     para = " ".join(corpus)
```



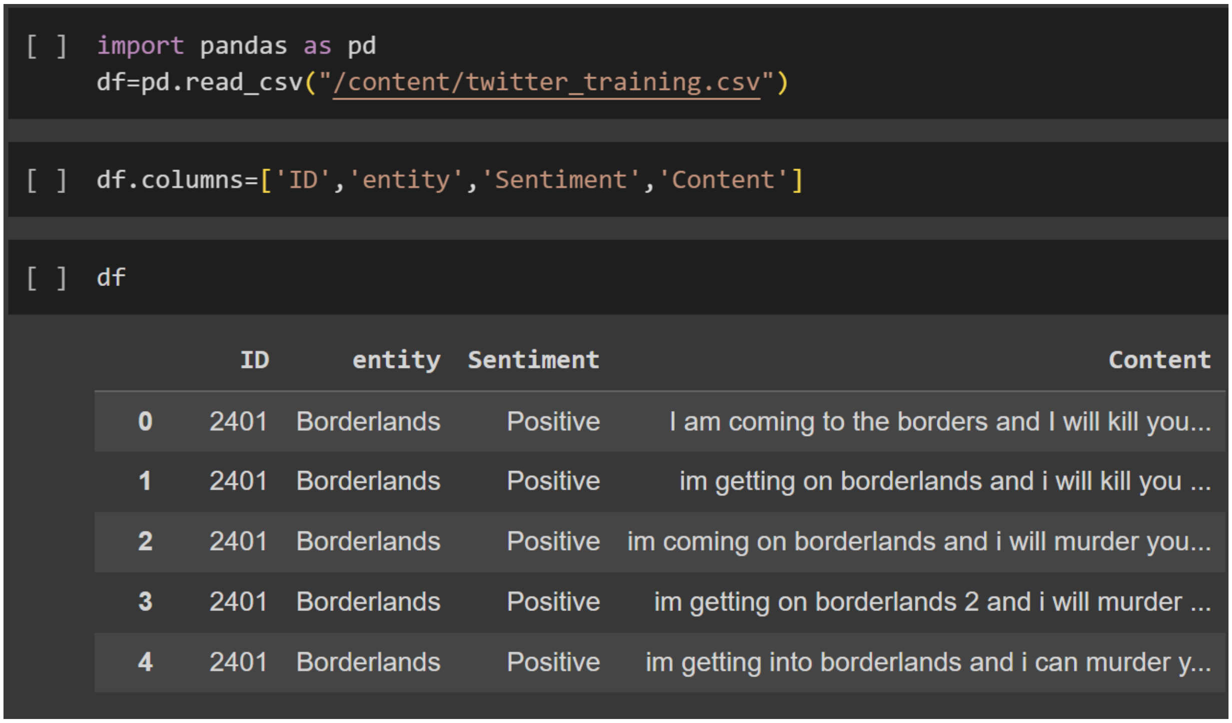
```
[27] corpus = tokenization(sentence)
     corpus = lowercase(corpus)
     corpus = num_to_words(corpus)
     corpus = remove_punctuation(corpus)
     remove_whitespaces(corpus)
     remove_stopwords(corpus)
     Count_Word_Frequency(corpus)
     counter_list = Count_Word_Frequency(corpus)
     ps_list = porter_stemmer(corpus)
     ls_list = porter_stemmer(corpus)
     lm_list = lemmatization(corpus)
```

Final Corpus, After all transformation applied



```
[30] corpus[:10]

     ['gener',
      'random',
      'paragraph',
      'can',
      'be',
      'an',
      'excel',
      'way',
      'for',
      'writer']
```

Performing All the Steps on Twitter Dataset

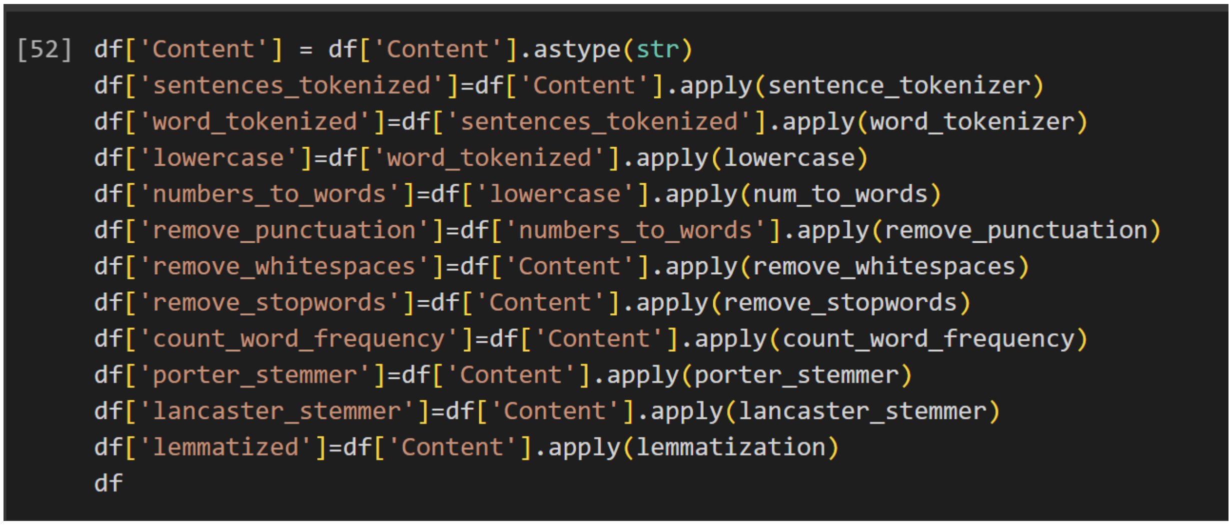


```
[ ]  import pandas as pd
     df=pd.read_csv("/content/twitter_training.csv")
```

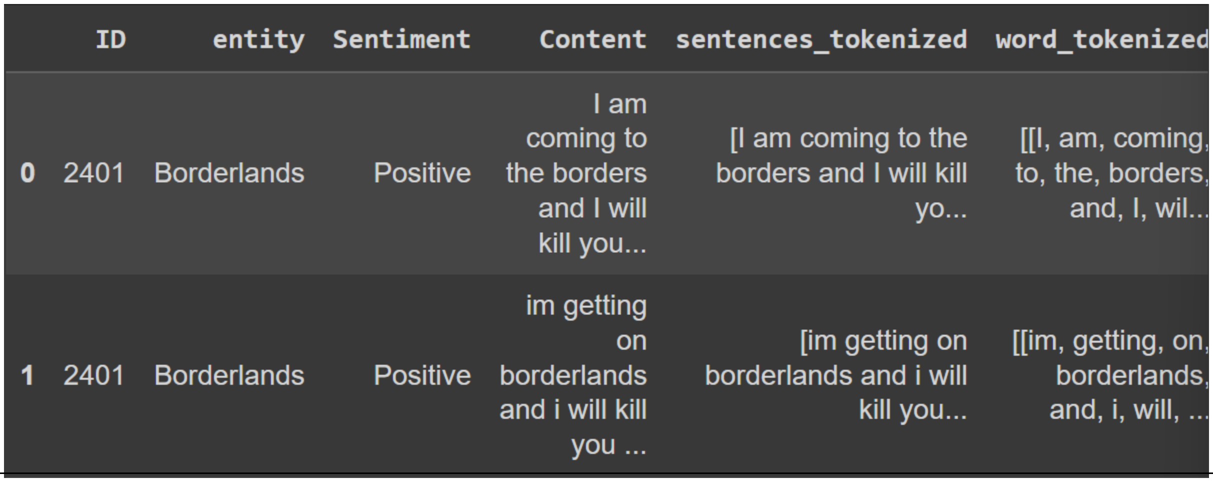
```
[ ]  df.columns=['ID','entity','Sentiment','Content']
```

```
[ ]  df
```

	ID	entity	Sentiment	Content
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...



```
[52] df['Content'] = df['Content'].astype(str)
     df['sentences_tokenized']=df['Content'].apply(sentence_tokenizer)
     df['word_tokenized']=df['sentences_tokenized'].apply(word_tokenizer)
     df['lowercase']=df['word_tokenized'].apply(lowercase)
     df['numbers_to_words']=df['lowercase'].apply(num_to_words)
     df['remove_punctuation']=df['numbers_to_words'].apply(remove_punctuation)
     df['remove_whitespaces']=df['Content'].apply(remove_whitespaces)
     df['remove_stopwords']=df['Content'].apply(remove_stopwords)
     df['count_word_frequency']=df['Content'].apply(count_word_frequency)
     df['porter_stemmer']=df['Content'].apply(porter_stemmer)
     df['lancaster_stemmer']=df['Content'].apply(lancaster_stemmer)
     df['lemmatized']=df['Content'].apply(lemmatization)
     df
```



	ID	entity	Sentiment	Content	sentences_tokenized	word_tokenized
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	[I am coming to the borders and I will kill yo...	[[I, am, coming, to, the, borders, and, I, wil...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	[im getting on borderlands and i will kill you...	[[im, getting, on, borderlands, and, i, will, ...