CSE6060

Statistical Natural Language Processing

Activity 1

Name : Kavianand G

Reg. No. : 19MAI0050

Date : 13 – June – 2020

Explore - NLTK and Corpus

In [1]:

```
#Importing necessasry packages
import nltk
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem import RegexpStemmer
from nltk.stem import SnowballStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk import pos_tag
```

Brown Corpus

```
    The Brown Corpus was the first million-word electronic corpus of English, crea
ted in 1961 at Brown University. This corpus contains text from 500 sources, and t
he sources have been categorized by genre, such as news, editorial, and so on.
```

In [2]:

```
# here I (Kavianand) used brown corpus
from nltk.corpus import brown
```

In [3]:

```
#viewing raw data from brown corpus
print(brown.raw()[:10])
print("-" *100)
print(brown.raw()[:10000])
```

```


	The/at
----------------------------------------------------------------------------
--------------------------



	The/at Fulton/np-tl County/nn-tl Grand/jj-tl Jury/nn-tl said/vbd F
riday/nr an/at investigation/nn of/in Atlanta's/np$ recent/jj primary/nn e
lection/nn produced/vbd ``/`` no/at evidence/nn ''/'' that/cs any/dti irre
gularities/nns took/vbd place/nn ./.


	The/at jury/nn further/rbr said/vbd in/in term-end/nn presentment
s/nns that/cs the/at City/nn-tl Executive/jj-tl Committee/nn-tl ,/, which/
wdt had/hvd over-all/jj charge/nn of/in the/at election/nn ,/, ``/`` deser
ves/vbz the/at praise/nn and/cc thanks/nns of/in the/at City/nn-tl of/in-t
l Atlanta/np-tl ''/'' for/in the/at manner/nn in/in which/wdt the/at elect
ion/nn was/bedz conducted/vbn ./.
```

In [4]:

```
#print number of characters in Brown Corpus
print("Characters   : ",len(brown.raw()))
#print number of words in Brown Corpus
print("Words        : ",len(brown.words()))
#print the number of sentences in brown corpus
print("Sentences    : ",len(brown.sents()))
```

```
Characters   :  9964284
Words        :  1161192
Sentences    :  57340
```

In [5]:

```
print("No. of Categories : ",len(brown.categories()))
#List the categories in brown corpus
print(brown.categories())
```

```
No. of Categories :  15
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbi
es', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews',
'romance', 'science_fiction']
```

In [6]:

```
#print first 50 words from brown corpus
print(brown.words()[:50])
```

```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'invest
igation', 'of', "Atlanta's", 'recent', 'primary', 'election', 'produced', '`
`', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'plac
e', '.', 'The', 'jury', 'further', 'said', 'in', 'term-end', 'presentments',
'that', 'the', 'City', 'Executive', 'Committee', ',', 'which', 'had', 'over-
all', 'charge', 'of', 'the', 'election', ',', '``', 'deserves', 'the', 'prai
se']
```

In [7]:

```
#print first 5 sentences from brown corpus
# the sentences are split into words
print(brown.sents()[:5])
```

```
[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'inves
tigation', 'of', "Atlanta's", 'recent', 'primary', 'election', 'produced',
'``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'plac
e', '.'], ['The', 'jury', 'further', 'said', 'in', 'term-end', 'presentment
s', 'that', 'the', 'City', 'Executive', 'Committee', ',', 'which', 'had', 'o
ver-all', 'charge', 'of', 'the', 'election', ',', '``', 'deserves', 'the',
'praise', 'and', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta', "''", 'fo
r', 'the', 'manner', 'in', 'which', 'the', 'election', 'was', 'conducted',
'.'], ['The', 'September-October', 'term', 'jury', 'had', 'been', 'charged',
'by', 'Fulton', 'Superior', 'Court', 'Judge', 'Durwood', 'Pye', 'to', 'inves
tigate', 'reports', 'of', 'possible', '``', 'irregularities', "''", 'in', 't
he', 'hard-fought', 'primary', 'which', 'was', 'won', 'by', 'Mayor-nominat
e', 'Ivan', 'Allen', 'Jr.', '.'], ['``', 'Only', 'a', 'relative', 'handful',
'of', 'such', 'reports', 'was', 'received', "''", ',', 'the', 'jury', 'sai
d', ',', '``', 'considering', 'the', 'widespread', 'interest', 'in', 'the',
'election', ',', 'the', 'number', 'of', 'voters', 'and', 'the', 'size', 'o
f', 'this', 'city', "''", '.'], ['The', 'jury', 'said', 'it', 'did', 'find',
'that', 'many', 'of', "Georgia's", 'registration', 'and', 'election', 'law
s', '``', 'are', 'outmoded', 'or', 'inadequate', 'and', 'often', 'ambiguou
s', "''", '.']]
```

In [8]:

```
#print 2 paragraphs from brown corpus
print(brown.paras()[:2])
```

```
[[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'inve
stigation', 'of', "Atlanta's", 'recent', 'primary', 'election', 'produced',
'``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'plac
e', '.']], [['The', 'jury', 'further', 'said', 'in', 'term-end', 'presentmen
ts', 'that', 'the', 'City', 'Executive', 'Committee', ',', 'which', 'had',
'over-all', 'charge', 'of', 'the', 'election', ',', '``', 'deserves', 'the',
'praise', 'and', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta', "''", 'fo
r', 'the', 'manner', 'in', 'which', 'the', 'election', 'was', 'conducted',
'.']]]
```

In [9]:

```python
for sent in brown.sents()[:3]: # First 3 sentences.
    text = (' '.join(sent))
    print(text)
```

```
The Fulton County Grand Jury said Friday an investigation of Atlanta's recen
t primary election produced `` no evidence '' that any irregularities took p
lace .
The jury further said in term-end presentments that the City Executive Commi
ttee , which had over-all charge of the election , `` deserves the praise an
d thanks of the City of Atlanta '' for the manner in which the election was
conducted .
The September-October term jury had been charged by Fulton Superior Court Ju
dge Durwood Pye to investigate reports of possible `` irregularities '' in t
he hard-fought primary which was won by Mayor-nominate Ivan Allen Jr. .
```

In [10]:

```python
#print tagged words from brown corpus
print(brown.tagged_words()[:50])
```

```
[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'), ('Grand', 'JJ-T
L'), ('Jury', 'NN-TL'), ('said', 'VBD'), ('Friday', 'NR'), ('an', 'AT'), ('i
nvestigation', 'NN'), ('of', 'IN'), ("Atlanta's", 'NP$'), ('recent', 'JJ'),
('primary', 'NN'), ('election', 'NN'), ('produced', 'VBD'), ('``', '``'),
('no', 'AT'), ('evidence', 'NN'), ("''", "''"), ('that', 'CS'), ('any', 'DT
I'), ('irregularities', 'NNS'), ('took', 'VBD'), ('place', 'NN'), ('.',
'.'), ('The', 'AT'), ('jury', 'NN'), ('further', 'RBR'), ('said', 'VBD'),
('in', 'IN'), ('term-end', 'NN'), ('presentments', 'NNS'), ('that', 'CS'),
('the', 'AT'), ('City', 'NN-TL'), ('Executive', 'JJ-TL'), ('Committee', 'NN-
TL'), (',', ','), ('which', 'WDT'), ('had', 'HVD'), ('over-all', 'JJ'), ('ch
arge', 'NN'), ('of', 'IN'), ('the', 'AT'), ('election', 'NN'), (',', ','),
('``', '``'), ('deserves', 'VBZ'), ('the', 'AT'), ('praise', 'NN')]
```

In [11]:

```python
#print tagged sentences from brown corpus
print(brown.tagged_sents()[:50])

```

```
[[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'), ('Grand', 'JJ-T
L'), ('Jury', 'NN-TL'), ('said', 'VBD'), ('Friday', 'NR'), ('an', 'AT'),
('investigation', 'NN'), ('of', 'IN'), ("Atlanta's", 'NP$'), ('recent', 'J
J'), ('primary', 'NN'), ('election', 'NN'), ('produced', 'VBD'), ('``', '`
`'), ('no', 'AT'), ('evidence', 'NN'), ("''", "''"), ('that', 'CS'), ('an
y', 'DTI'), ('irregularities', 'NNS'), ('took', 'VBD'), ('place', 'NN'),
('.', '.')], [('The', 'AT'), ('jury', 'NN'), ('further', 'RBR'), ('said',
'VBD'), ('in', 'IN'), ('term-end', 'NN'), ('presentments', 'NNS'), ('tha
t', 'CS'), ('the', 'AT'), ('City', 'NN-TL'), ('Executive', 'JJ-TL'), ('Com
mittee', 'NN-TL'), (',', ','), ('which', 'WDT'), ('had', 'HVD'), ('over-al
l', 'JJ'), ('charge', 'NN'), ('of', 'IN'), ('the', 'AT'), ('election', 'N
N'), (',', ','), ('``', '``'), ('deserves', 'VBZ'), ('the', 'AT'), ('prais
e', 'NN'), ('and', 'CC'), ('thanks', 'NNS'), ('of', 'IN'), ('the', 'AT'),
('City', 'NN-TL'), ('of', 'IN-TL'), ('Atlanta', 'NP-TL'), ("''", "''"),
('for', 'IN'), ('the', 'AT'), ('manner', 'NN'), ('in', 'IN'), ('which', 'W
DT'), ('the', 'AT'), ('election', 'NN'), ('was', 'BEDZ'), ('conducted', 'V
BN'), ('.', '.')], [('The', 'AT'), ('September-October', 'NP'), ('term',
'NN'), ('jury', 'NN'), ('had', 'HVD'), ('been', 'BEN'), ('charged', 'VB
N'), ('by', 'IN'), ('Fulton', 'NP-TL'), ('Superior', 'JJ-TL'), ('Court',
```

Frequency Distribution

In [12]:

```python
text = brown.words(categories='reviews')
fdist = nltk.FreqDist(w.lower() for w in text)
modals = [ 'good', 'bad', 'average',
          'can', 'could', 'may', 'might', 'must', 'will']

for m in modals:
    print(m + ':', fdist[m], end=' ')
    print("\n")
```

```
good: 44

bad: 5

average: 1

can: 45

could: 40

may: 47

might: 26

must: 19

will: 61
```

Conditional Frequency Distribution

In [13]:

```
cfd = nltk.ConditionalFreqDist((genre, word)
           for genre in brown.categories()
           for word in brown.words(categories=genre))
genres = ['news','reviews', 'religion', 'hobbies', 'science_fiction', 'romance', 'humor
modals = ['Good','can', 'could', 'may', 'might', 'must', 'will']
cfd.tabulate(conditions=genres, samples=modals)
```

```
                Good   can could   may might  must  will
           news    1    93    86    66    38    50   389
        reviews    2    45    40    45    26    19    58
       religion    1    82    59    78    12    54    71
        hobbies    7   268    58   131    22    83   264
science_fiction    1    16    49     4    12     8    16
        romance    4    74   193    11    51    45    43
          humor    1    16    30     8     8     9    13
```

In [ ]:

```

```

Gutenberg Corpus

In [14]:

```
from nltk.corpus import gutenberg
```

In [15]:

```
#List of files in Gutenberg corpus
gutenberg.fileids()
```

Out[15]:

```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

In [16]:

```
print("No. of Words :" ,len(gutenberg.words('shakespeare-caesar.txt')))
print(gutenberg.words(fileids='shakespeare-caesar.txt')[:100])
```

```
No. of Words : 25833
['[', 'The', 'Tragedie', 'of', 'Julius', 'Caesar', 'by', 'William', 'Shakesp
eare', '1599', ']', 'Actus', 'Primus', '.', 'Scoena', 'Prima', '.', 'Enter',
'Flauius', ',', 'Murellus', ',', 'and', 'certaine', 'Commoners', 'ouer', 'th
e', 'Stage', '.', 'Flauius', '.', 'Hence', ':', 'home', 'you', 'idle', 'Crea
tures', ',', 'get', 'you', 'home', ':', 'Is', 'this', 'a', 'Holiday', '?',
'What', ',', 'know', 'you', 'not', '(', 'Being', 'Mechanicall', ')', 'you',
'ought', 'not', 'walke', 'Vpon', 'a', 'labouring', 'day', ',', 'without', 't
he', 'signe', 'Of', 'your', 'Profession', '?', 'Speake', ',', 'what', 'Trad
e', 'art', 'thou', '?', 'Car', '.', 'Why', 'Sir', ',', 'a', 'Carpenter', 'Mu
r', '.', 'Where', 'is', 'thy', 'Leather', 'Apron', ',', 'and', 'thy', 'Rul
e', '?', 'What', 'dost']
```

In [17]:

```
for fileid in gutenberg.fileids():
    print(gutenberg.raw(fileids='shakespeare-caesar.txt')[:])
```

```
[The Tragedie of Julius Caesar by William Shakespeare 1599]


Actus Primus. Scoena Prima.

Enter Flauius, Murellus, and certaine Commoners ouer the Stage.

  Flauius. Hence: home you idle Creatures, get you home:
Is this a Holiday? What, know you not
(Being Mechanicall) you ought not walke
Vpon a labouring day, without the signe
Of your Profession? Speake, what Trade art thou?
  Car. Why Sir, a Carpenter

   Mur. Where is thy Leather Apron, and thy Rule?
What dost thou with thy best Apparrell on?
You sir, what Trade are you?
  Cobl. Truely Sir, in respect of a fine Workman, I am
but as you would say, a Cobler
```

Frequency Distribution

In [18]:

```
text = gutenberg.words('shakespeare-caesar.txt')
fdist = nltk.FreqDist(w.lower() for w in text)
modals = [ 'caesar', 'julius', 'cassius',
          'what', 'could', 'may', 'might', 'must', 'will']

for m in modals:
    print(m + ':', fdist[m], end=' ')
    print("\n")
```

```
caesar: 190

julius: 1

cassius: 85

what: 129

could: 18

may: 38

might: 13

must: 36

will: 163
```

Lexicons

In [19]:

```
from nltk.corpus import names, stopwords, words
```

In [20]:

```
words.fileids()
```

Out[20]:

```
['en', 'en-basic']
```

In [21]:

```
print("No. of Words :" ,len(words.words('en')))
print(words.words('en')[:100])
```

```
No. of Words : 235886
['A', 'a', 'aa', 'aal', 'aalii', 'aam', 'Aani', 'aardvark', 'aardwolf', 'Aar
on', 'Aaronic', 'Aaronical', 'Aaronite', 'Aaronitic', 'Aaru', 'Ab', 'aba',
'Ababdeh', 'Ababua', 'abac', 'abaca', 'abacate', 'abacay', 'abacinate', 'aba
cination', 'abaciscus', 'abacist', 'aback', 'abactinal', 'abactinally', 'aba
ction', 'abactor', 'abaculus', 'abacus', 'Abadite', 'abaff', 'abaft', 'abais
ance', 'abaiser', 'abaissed', 'abalienate', 'abalienation', 'abalone', 'Abam
a', 'abampere', 'abandon', 'abandonable', 'abandoned', 'abandonedly', 'aband
onee', 'abandoner', 'abandonment', 'Abanic', 'Abantes', 'abaptiston', 'Abara
mbo', 'Abaris', 'abarthrosis', 'abarticular', 'abarticulation', 'abas', 'aba
se', 'abased', 'abasedly', 'abasedness', 'abasement', 'abaser', 'Abasgi', 'a
bash', 'abashed', 'abashedly', 'abashedness', 'abashless', 'abashlessly', 'a
bashment', 'abasia', 'abasic', 'abask', 'Abassin', 'abastardize', 'abatabl
e', 'abate', 'abatement', 'abater', 'abatis', 'abatised', 'abaton', 'abato
r', 'abattoir', 'Abatua', 'abature', 'abave', 'abaxial', 'abaxile', 'abaze',
'abb', 'Abba', 'abbacomes', 'abbacy', 'Abbadide']
```

In [22]:

```
stopwords.fileids()
```

Out[22]:

```
['arabic',
 'azerbaijani',
 'danish',
 'dutch',
 'english',
 'finnish',
 'french',
 'german',
 'greek',
 'hungarian',
 'indonesian',
 'italian',
 'kazakh',
 'nepali',
 'norwegian',
 'portuguese',
 'romanian',
 'russian',
```

In [23]:

```
print("No. of Words :" ,len(stopwords.words('german')))
print(stopwords.words('german'))
```

```
No. of Words : 232
['aber', 'alle', 'allem', 'allen', 'aller', 'alles', 'als', 'also', 'am', 'a
n', 'ander', 'andere', 'anderem', 'anderen', 'anderer', 'anderes', 'anderm',
'andern', 'anderr', 'anders', 'auch', 'auf', 'aus', 'bei', 'bin', 'bis', 'bi
st', 'da', 'damit', 'dann', 'der', 'den', 'des', 'dem', 'die', 'das', 'das
s', 'daß', 'derselbe', 'derselben', 'denselben', 'desselben', 'demselben',
'dieselbe', 'dieselben', 'dasselbe', 'dazu', 'dein', 'deine', 'deinem', 'dei
nen', 'deiner', 'deines', 'denn', 'derer', 'dessen', 'dich', 'dir', 'du', 'd
ies', 'diese', 'diesem', 'diesen', 'dieser', 'dieses', 'doch', 'dort', 'durc
h', 'ein', 'eine', 'einem', 'einen', 'einer', 'eines', 'einig', 'einige', 'e
inigem', 'einigen', 'einiger', 'einiges', 'einmal', 'er', 'ihn', 'ihm', 'e
s', 'etwas', 'euer', 'eure', 'eurem', 'euren', 'eurer', 'eures', 'für', 'geg
en', 'gewesen', 'hab', 'habe', 'haben', 'hat', 'hatte', 'hatten', 'hier', 'h
in', 'hinter', 'ich', 'mich', 'mir', 'ihr', 'ihre', 'ihrem', 'ihren', 'ihre
r', 'ihres', 'euch', 'im', 'in', 'indem', 'ins', 'ist', 'jede', 'jedem', 'je
den', 'jeder', 'jedes', 'jene', 'jenem', 'jenen', 'jener', 'jenes', 'jetzt',
'kann', 'kein', 'keine', 'keinem', 'keinen', 'keiner', 'keines', 'können',
'könnte', 'machen', 'man', 'manche', 'manchem', 'manchen', 'mancher', 'manch
es', 'mein', 'meine', 'meinem', 'meinen', 'meiner', 'meines', 'mit', 'muss',
'musste', 'nach', 'nicht', 'nichts', 'noch', 'nun', 'nur', 'ob', 'oder', 'oh
ne', 'sehr', 'sein', 'seine', 'seinem', 'seinen', 'seiner', 'seines', 'selbs
t', 'sich', 'sie', 'ihnen', 'sind', 'so', 'solche', 'solchem', 'solchen', 's
olcher', 'solches', 'soll', 'sollte', 'sondern', 'sonst', 'über', 'um', 'un
d', 'uns', 'unsere', 'unserem', 'unseren', 'unser', 'unseres', 'unter', 'vie
l', 'vom', 'von', 'vor', 'während', 'war', 'waren', 'warst', 'was', 'weg',
'weil', 'weiter', 'welche', 'welchem', 'welchen', 'welcher', 'welches', 'wen
n', 'werde', 'werden', 'wie', 'wieder', 'will', 'wir', 'wird', 'wirst', 'w
o', 'wollen', 'wollte', 'würde', 'würden', 'zu', 'zum', 'zur', 'zwar', 'zwis
chen']
```

In [24]:

```
print("No. of Words :" ,len(stopwords.words('english')))
print(stopwords.words('english'))
```

```
No. of Words : 179
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'r
e", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'i
t', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselve
s', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'tho
se', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has',
'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'bu
t', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for',
'with', 'about', 'against', 'between', 'into', 'through', 'during', 'befor
e', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'o
n', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'the
re', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'mo
re', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'sa
me', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "d
on't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y',
'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
oesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "is
n't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 's
han', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "were
n't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [25]:

```
names.fileids()
```

Out[25]:

```
['female.txt', 'male.txt']
```

In [26]:

```
print("No. of Words :" ,len(names.words('male.txt')))
print(names.words('male.txt')[:100])

```

```
No. of Words : 2943
['Aamir', 'Aaron', 'Abbey', 'Abbie', 'Abbot', 'Abbott', 'Abby', 'Abdel', 'Ab
dul', 'Abdulkarim', 'Abdullah', 'Abe', 'Abel', 'Abelard', 'Abner', 'Abraha
m', 'Abram', 'Ace', 'Adair', 'Adam', 'Adams', 'Addie', 'Adger', 'Aditya', 'A
dlai', 'Adnan', 'Adolf', 'Adolfo', 'Adolph', 'Adolphe', 'Adolpho', 'Adolphu
s', 'Adrian', 'Adrick', 'Adrien', 'Agamemnon', 'Aguinaldo', 'Aguste', 'Agust
in', 'Aharon', 'Ahmad', 'Ahmed', 'Ahmet', 'Ajai', 'Ajay', 'Al', 'Alaa', 'Ala
in', 'Alan', 'Alasdair', 'Alastair', 'Albatros', 'Albert', 'Alberto', 'Albre
cht', 'Alden', 'Aldis', 'Aldo', 'Aldric', 'Aldrich', 'Aldus', 'Aldwin', 'Ale
c', 'Aleck', 'Alejandro', 'Aleks', 'Aleksandrs', 'Alessandro', 'Alex', 'Alex
ander', 'Alexei', 'Alexis', 'Alf', 'Alfie', 'Alfonse', 'Alfonso', 'Alfonzo',
'Alford', 'Alfred', 'Alfredo', 'Algernon', 'Ali', 'Alic', 'Alister', 'Alix',
'Allah', 'Allan', 'Allen', 'Alley', 'Allie', 'Allin', 'Allyn', 'Alonso', 'Al
onzo', 'Aloysius', 'Alphonse', 'Alphonso', 'Alston', 'Alton', 'Alvin']
```

In [27]:

```
if "George" in names.words('male.txt'):
    print("True")
else:
    print("False")
```

```
True
```

Explore - Stemming, Tokenizing, POS Tagging

In [28]:

```
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem import RegexpStemmer
from nltk.stem import SnowballStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk import pos_tag
```

In [29]:

```
porter = PorterStemmer()
lancaster=LancasterStemmer()
#A list of words to be stemmed
word_list = ["friend", "friendship", "friends", "friendships","stabil","destabilize","r
print("{0:20}{1:20}{2:20}".format("Word","Porter Stemmer","lancaster Stemmer"))
print("-" *60)
for word in word_list:
    print("{0:20}{1:20}{2:20}".format(word,porter.stem(word),lancaster.stem(word)))
```

```
Word                Porter Stemmer      lancaster Stemmer
------------------------------------------------------------
friend              friend              friend
friendship          friendship          friend
friends             friend              friend
friendships         friendship          friend
stabil              stabil              stabl
destabilize         destabil            dest
misunderstanding    misunderstand       misunderstand
railroad            railroad            railroad
moonlight           moonlight           moonlight
football            footbal             footbal
```

Using Shakespeare ceasar text file for stemming, Tokenizing

In [30]:

```python
abc = gutenberg.raw('shakespeare-caesar.txt')
print(abc)
```

```
[The Tragedie of Julius Caesar by William Shakespeare 1599]


Actus Primus. Scoena Prima.

Enter Flauius, Murellus, and certaine Commoners ouer the Stage.

  Flauius. Hence: home you idle Creatures, get you home:
Is this a Holiday? What, know you not
(Being Mechanicall) you ought not walke
Vpon a labouring day, without the signe
Of your Profession? Speake, what Trade art thou?
  Car. Why Sir, a Carpenter

   Mur. Where is thy Leather Apron, and thy Rule?
What dost thou with thy best Apparrell on?
You sir, what Trade are you?
  Cobl. Truely Sir, in respect of a fine Workman, I am
but as you would say, a Cobler
```

Using Shakespeare ceasar text file for stemming, Tokenizing

Sentence Tokenization using #sent_tokenize option

In [31]:

```python
sentence = sent_tokenize(abc)
print(sentence)
```

```
['[The Tragedie of Julius Caesar by William Shakespeare 1599]\n\n\nActus P
rimus.', 'Scoena Prima.', 'Enter Flauius, Murellus, and certaine Commoners
ouer the Stage.', 'Flauius.', 'Hence: home you idle Creatures, get you hom
e:\nIs this a Holiday?', 'What, know you not\n(Being Mechanicall) you ough
t not walke\nVpon a labouring day, without the signe\nOf your Professio
n?', 'Speake, what Trade art thou?', 'Car.', 'Why Sir, a Carpenter\n\n   M
ur.', 'Where is thy Leather Apron, and thy Rule?', 'What dost thou with th
y best Apparrell on?', 'You sir, what Trade are you?', 'Cobl.', 'Truely Si
r, in respect of a fine Workman, I am\nbut as you would say, a Cobler\n\n
Mur.', 'But what Trade art thou?', 'Answer me directly\n\n   Cob.', 'A Tra
de Sir, that I hope I may vse, with a safe\nConscience, which is indeed Si
r, a Mender of bad soules\n\n   Fla. What Trade thou knaue?', 'Thou naught
y knaue,\nwhat Trade?', 'Cobl.', 'Nay I beseech you Sir, be not out with m
e: yet\nif you be out Sir, I can mend you\n\n   Mur.', "What mean'st thou
by that?", 'Mend mee, thou\nsawcy Fellow?', 'Cob.', 'Why sir, Cobble you\n
\n   Fla. Thou art a Cobler, art thou?', 'Cob.', 'Truly sir, all that I li
ue by, is with the Aule: I\nmeddle with no Tradesmans matters, nor womens
matters;\nbut withal I am indeed Sir, a Surgeon to old shooes:\nwhen they
are in great danger, I recouer them.', 'As proper\nmen as euer trod vpon N
```

Word Tokenization using #word_tokenize option

In [32]:

```
words=word_tokenize(abc)
print(words)
```

```
['[', 'The', 'Tragedie', 'of', 'Julius', 'Caesar', 'by', 'William', 'Shake
speare', '1599', ']', 'Actus', 'Primus', '.', 'Scoena', 'Prima', '.', 'Ent
er', 'Flauius', ',', 'Murellus', ',', 'and', 'certaine', 'Commoners', 'oue
r', 'the', 'Stage', '.', 'Flauius', '.', 'Hence', ':', 'home', 'you', 'idl
e', 'Creatures', ',', 'get', 'you', 'home', ':', 'Is', 'this', 'a', 'Holid
ay', '?', 'What', ',', 'know', 'you', 'not', '(', 'Being', 'Mechanicall',
')', 'you', 'ought', 'not', 'walke', 'Vpon', 'a', 'labouring', 'day', ',',
'without', 'the', 'signe', 'Of', 'your', 'Profession', '?', 'Speake', ',',
'what', 'Trade', 'art', 'thou', '?', 'Car', '.', 'Why', 'Sir', ',', 'a',
'Carpenter', 'Mur', '.', 'Where', 'is', 'thy', 'Leather', 'Apron', ',', 'a
nd', 'thy', 'Rule', '?', 'What', 'dost', 'thou', 'with', 'thy', 'best', 'A
pparrell', 'on', '?', 'You', 'sir', ',', 'what', 'Trade', 'are', 'you',
'?', 'Cobl', '.', 'Truely', 'Sir', ',', 'in', 'respect', 'of', 'a', 'fin
e', 'Workman', ',', 'I', 'am', 'but', 'as', 'you', 'would', 'say', ',',
'a', 'Cobler', 'Mur', '.', 'But', 'what', 'Trade', 'art', 'thou', '?', 'An
swer', 'me', 'directly', 'Cob', '.', 'A', 'Trade', 'Sir', ',', 'that',
'I', 'hope', 'I', 'may', 'vse', ',', 'with', 'a', 'safe', 'Conscience',
',', 'which', 'is', 'indeed', 'Sir', ',', 'a', 'Mender', 'of', 'bad', 'sou
les', 'Fla.', 'What', 'Trade', 'thou', 'knaue', '?', 'Thou', 'naughty', 'k
```

Tagged using #pos_tag option

In [33]:

```
tagged = pos_tag(words)
print(tagged)
```

```
[('[', 'IN'), ('The', 'DT'), ('Tragedie', 'NNP'), ('of', 'IN'), ('Julius',
'NNP'), ('Caesar', 'NNP'), ('by', 'IN'), ('William', 'NNP'), ('Shakespear
e', 'NNP'), ('1599', 'CD'), (']', 'NNP'), ('Actus', 'NNP'), ('Primus', 'NN
P'), ('.', '.'), ('Scoena', 'NNP'), ('Prima', 'NNP'), ('.', '.'), ('Ente
r', 'NNP'), ('Flauius', 'NNP'), (',', ','), ('Murellus', 'NNP'), (',',
','), ('and', 'CC'), ('certaine', 'NN'), ('Commoners', 'NNP'), ('ouer', 'V
BZ'), ('the', 'DT'), ('Stage', 'NN'), ('.', '.'), ('Flauius', 'NNP'),
('.', '.'), ('Hence', 'NN'), (':', ':'), ('home', 'NN'), ('you', 'PRP'),
('idle', 'JJ'), ('Creatures', 'NNS'), (',', ','), ('get', 'VBP'), ('you',
'PRP'), ('home', 'NN'), (':', ':'), ('Is', 'VBZ'), ('this', 'DT'), ('a',
'DT'), ('Holiday', 'NNP'), ('?', '.'), ('What', 'WP'), (',', ','), ('kno
w', 'VBP'), ('you', 'PRP'), ('not', 'RB'), ('(', '('), ('Being', 'VBG'),
('Mechanicall', 'NNP'), (')', ')'), ('you', 'PRP'), ('ought', 'MD'), ('no
t', 'RB'), ('walke', 'VB'), ('Vpon', 'NNP'), ('a', 'DT'), ('labouring', 'J
J'), ('day', 'NN'), (',', ','), ('without', 'IN'), ('the', 'DT'), ('sign
e', 'NN'), ('Of', 'IN'), ('your', 'PRP$'), ('Profession', 'NN'), ('?',
'.'), ('Speake', 'NNP'), (',', ','), ('what', 'WP'), ('Trade', 'NNP'), ('a
rt', 'NN'), ('thou', 'NN'), ('?', '.'), ('Car', 'NNP'), ('.', '.'), ('Wh
y', 'WRB'), ('Sir', 'NNP'), (',', ','), ('a', 'DT'), ('Carpenter', 'NNP'),
```

Apply Stemming on shakespeare work

In [34]:

```python
porter = PorterStemmer()
lancaster=LancasterStemmer()
print("{0:20}{1:20}{2:20}".format("Word","Porter Stemmer","lancaster Stemmer"))
print("-" *60)
for word in words:
    print("{0:20}{1:20}{2:20}".format(word,porter.stem(word),lancaster.stem(word)))
```

```
Word                Porter Stemmer      lancaster Stemmer
------------------------------------------------------------
[                   [                   [
The                 the                 the
Tragedie            tragedi             tragedy
of                  of                  of
Julius              juliu               juli
Caesar              caesar              caes
by                  by                  by
William             william             william
Shakespeare         shakespear          shakespear
1599                1599                1599
]                   ]                   ]
Actus               actu                act
Primus              primu               prim
.                   .                   .
Scoena              scoena              scoen
Prima               prima               prim
.                   .                   .
```

In [35]:

```python
token_words=word_tokenize(abc)
stem_sentence=[]
for word in token_words:
    stem_sentence.append(porter.stem(word))
    stem_sentence.append(" ")
#print(stem_sentence)
print( "".join(stem_sentence))
```

```
[ the tragedi of juliu caesar by william shakespear 1599 ] actu primu . sc
oena prima . enter flauiu , murellu , and certain common ouer the stage .
flauiu . henc : home you idl creatur , get you home : Is thi a holiday ? w
hat , know you not ( be mechanical ) you ought not walk vpon a labour day
, without the sign Of your profess ? speak , what trade art thou ? car . w
hi sir , a carpent mur . where is thi leather apron , and thi rule ? what
dost thou with thi best apparrel on ? you sir , what trade are you ? cobl
. trueli sir , in respect of a fine workman , I am but as you would say ,
a cobler mur . but what trade art thou ? answer me directli cob . A trade
sir , that I hope I may vse , with a safe conscienc , which is inde sir ,
a mender of bad soul fla. what trade thou knaue ? thou naughti knaue , wha
t trade ? cobl . nay I beseech you sir , be not out with me : yet if you b
e out sir , I can mend you mur . what mean'st thou by that ? mend mee , th
ou sawci fellow ? cob . whi sir , cobbl you fla. thou art a cobler , art t
hou ? cob . truli sir , all that I liue by , is with the aul : I meddl wit
h no tradesman matter , nor women matter ; but withal I am inde sir , a su
rgeon to old shooe : when they are in great danger , I recouer them . As p
roper men as euer trod vpon neat leather , haue gone vpon my handy-work fl
a . but wherefor art not in thi shop to day ? whi do'st thou lead these me
```

Using Lemmatizer

In [36]:

```python
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

In [37]:

```python
for word in words:
    print("{0:20}{1:20}".format(word,lemmatizer.lemmatize(word)))
```

```
[                   [
The                 The
Tragedie            Tragedie
of                  of
Julius              Julius
Caesar              Caesar
by                  by
William             William
Shakespeare         Shakespeare
1599                1599
]                   ]
Actus               Actus
Primus              Primus
.                   .
Scoena              Scoena
Prima               Prima
.                   .
Enter               Enter
Flauius             Flauius
```

Inference on using Lemmatizer on shakespeare work

```
Not that of significance on fictional role play work. Even the Stemmer (Both porte
r and lancanster) are not of significant use, as the work has lot of poetric sente
nce formation. Let us explore it in normal webtext.
```

Using Stemming, Tokenizing on Wikipedia context of Julius Ceasar. (Normal English - No poetric touch)

In [38]:

```python
text = """Gaius Julius Caesar (12 July 100 BC – 15 March 44 BC), known simply as Julius
"""
```

In [39]:

```
sentence = sent_tokenize(text)
for sent in sentence:
    print(sent+ "\n")
```

```
Gaius Julius Caesar (12 July 100 BC – 15 March 44 BC), known simply as Juliu
s Caesar, was a Roman general and statesman who played a critical role in th
e events that led to the demise of the Roman Republic and the rise of the Ro
man Empire.

He was also a historian and author of Latin prose.

In 60 BC, Caesar, Crassus and Pompey formed the First Triumvirate, a politic
al alliance that dominated Roman politics for several years.

Their attempts to amass power as Populares were opposed by the Optimates wit
hin the Roman Senate, among them Cato the Younger with the frequent support
of Cicero.

Caesar rose to become one of the most powerful politicians in the Roman Repu
blic through a number of his accomplishments, notably his victories in the G
allic Wars, completed by 51 BC.

During this time, Caesar became the first Roman general to cross both the En
glish Channel and the Rhine River, when he built a bridge across the Rhine a
nd crossed the Channel to invade Britain.

Caesar's wars extended Rome's territory to Britain and past Gaul.

These achievements granted him unmatched military power and threatened to ec
lipse the standing of Pompey, who had realigned himself with the Senate afte
r the death of Crassus in 53 BC.

With the Gallic Wars concluded, the Senate ordered Caesar to step down from
his military command and return to Rome.

Leaving his command in Gaul meant losing his immunity from being charged as
a criminal for waging unsanctioned wars.

As a result, Caesar found himself with no other options but to cross the Rub
icon with the 13th Legion in 49 BC, leaving his province and illegally enter
ing Roman Italy under arms.

This began Caesar's civil war, and his victory in the war by 45 BC put him i
n an unrivaled position of power and influence.
```

In [40]:

```
words=word_tokenize(text)
print(words)
```

```
['Gaius', 'Julius', 'Caesar', '(', '12', 'July', '100', 'BC', '–', '15', 'Ma
rch', '44', 'BC', ')', ',', 'known', 'simply', 'as', 'Julius', 'Caesar',
',', 'was', 'a', 'Roman', 'general', 'and', 'statesman', 'who', 'played',
'a', 'critical', 'role', 'in', 'the', 'events', 'that', 'led', 'to', 'the',
'demise', 'of', 'the', 'Roman', 'Republic', 'and', 'the', 'rise', 'of', 'th
e', 'Roman', 'Empire', '.', 'He', 'was', 'also', 'a', 'historian', 'and', 'a
uthor', 'of', 'Latin', 'prose', '.', 'In', '60', 'BC', ',', 'Caesar', ',',
'Crassus', 'and', 'Pompey', 'formed', 'the', 'First', 'Triumvirate', ',',
'a', 'political', 'alliance', 'that', 'dominated', 'Roman', 'politics', 'fo
r', 'several', 'years', '.', 'Their', 'attempts', 'to', 'amass', 'power', 'a
s', 'Populares', 'were', 'opposed', 'by', 'the', 'Optimates', 'within', 'th
e', 'Roman', 'Senate', ',', 'among', 'them', 'Cato', 'the', 'Younger', 'wit
h', 'the', 'frequent', 'support', 'of', 'Cicero', '.', 'Caesar', 'rose', 't
o', 'become', 'one', 'of', 'the', 'most', 'powerful', 'politicians', 'in',
'the', 'Roman', 'Republic', 'through', 'a', 'number', 'of', 'his', 'accompli
shments', ',', 'notably', 'his', 'victories', 'in', 'the', 'Gallic', 'Wars',
',', 'completed', 'by', '51', 'BC', '.', 'During', 'this', 'time', ',', 'Cae
sar', 'became', 'the', 'first', 'Roman', 'general', 'to', 'cross', 'both',
'the', 'English', 'Channel', 'and', 'the', 'Rhine', 'River', ',', 'when', 'h
e', 'built', 'a', 'bridge', 'across', 'the', 'Rhine', 'and', 'crossed', 'th
e', 'Channel', 'to', 'invade', 'Britain', '.', 'Caesar', "'s", 'wars', 'exte
nded', 'Rome', "'s", 'territory', 'to', 'Britain', 'and', 'past', 'Gaul',
'.', 'These', 'achievements', 'granted', 'him', 'unmatched', 'military', 'po
wer', 'and', 'threatened', 'to', 'eclipse', 'the', 'standing', 'of', 'Pompe
y', ',', 'who', 'had', 'realigned', 'himself', 'with', 'the', 'Senate', 'aft
er', 'the', 'death', 'of', 'Crassus', 'in', '53', 'BC', '.', 'With', 'the',
'Gallic', 'Wars', 'concluded', ',', 'the', 'Senate', 'ordered', 'Caesar', 't
o', 'step', 'down', 'from', 'his', 'military', 'command', 'and', 'return',
'to', 'Rome', '.', 'Leaving', 'his', 'command', 'in', 'Gaul', 'meant', 'losi
ng', 'his', 'immunity', 'from', 'being', 'charged', 'as', 'a', 'criminal',
'for', 'waging', 'unsanctioned', 'wars', '.', 'As', 'a', 'result', ',', 'Cae
sar', 'found', 'himself', 'with', 'no', 'other', 'options', 'but', 'to', 'cr
oss', 'the', 'Rubicon', 'with', 'the', '13th', 'Legion', 'in', '49', 'BC',
',', 'leaving', 'his', 'province', 'and', 'illegally', 'entering', 'Roman',
'Italy', 'under', 'arms', '.', 'This', 'began', 'Caesar', "'s", 'civil', 'wa
r', ',', 'and', 'his', 'victory', 'in', 'the', 'war', 'by', '45', 'BC', 'pu
t', 'him', 'in', 'an', 'unrivaled', 'position', 'of', 'power', 'and', 'influ
ence', '.']
```

In [41]:

```
tagged = pos_tag(words)
print(tagged)
```

```
[('Gaius', 'NNP'), ('Julius', 'NNP'), ('Caesar', 'NNP'), ('(', '('), ('12',
'CD'), ('July', 'NNP'), ('100', 'CD'), ('BC', 'NNP'), ('–', '$'), ('15', 'C
D'), ('March', 'NNP'), ('44', 'CD'), ('BC', 'NNP'), (')', ')'), (',', ','),
('known', 'VBN'), ('simply', 'RB'), ('as', 'IN'), ('Julius', 'NNP'), ('Caesa
r', 'NNP'), (',', ','), ('was', 'VBD'), ('a', 'DT'), ('Roman', 'NNP'), ('gen
eral', 'JJ'), ('and', 'CC'), ('statesman', 'NN'), ('who', 'WP'), ('played',
'VBD'), ('a', 'DT'), ('critical', 'JJ'), ('role', 'NN'), ('in', 'IN'), ('th
e', 'DT'), ('events', 'NNS'), ('that', 'WDT'), ('led', 'VBD'), ('to', 'TO'),
('the', 'DT'), ('demise', 'NN'), ('of', 'IN'), ('the', 'DT'), ('Roman', 'NN
P'), ('Republic', 'NNP'), ('and', 'CC'), ('the', 'DT'), ('rise', 'NN'), ('o
f', 'IN'), ('the', 'DT'), ('Roman', 'NNP'), ('Empire', 'NNP'), ('.', '.'),
('He', 'PRP'), ('was', 'VBD'), ('also', 'RB'), ('a', 'DT'), ('historian', 'J
J'), ('and', 'CC'), ('author', 'NN'), ('of', 'IN'), ('Latin', 'NNP'), ('pros
e', 'NN'), ('.', '.'), ('In', 'IN'), ('60', 'CD'), ('BC', 'NNP'), (',',
','), ('Caesar', 'NNP'), (',', ','), ('Crassus', 'NNP'), ('and', 'CC'), ('Po
mpey', 'NNP'), ('formed', 'VBD'), ('the', 'DT'), ('First', 'NNP'), ('Triumvi
rate', 'NNP'), (',', ','), ('a', 'DT'), ('political', 'JJ'), ('alliance', 'N
N'), ('that', 'WDT'), ('dominated', 'VBD'), ('Roman', 'NNP'), ('politics',
'NNS'), ('for', 'IN'), ('several', 'JJ'), ('years', 'NNS'), ('.', '.'), ('Th
eir', 'PRP$'), ('attempts', 'NNS'), ('to', 'TO'), ('amass', 'VB'), ('power',
'NN'), ('as', 'IN'), ('Populares', 'NNS'), ('were', 'VBD'), ('opposed', 'VB
N'), ('by', 'IN'), ('the', 'DT'), ('Optimates', 'NNP'), ('within', 'IN'),
('the', 'DT'), ('Roman', 'NNP'), ('Senate', 'NNP'), (',', ','), ('among', 'I
N'), ('them', 'PRP'), ('Cato', 'NNP'), ('the', 'DT'), ('Younger', 'NNP'),
('with', 'IN'), ('the', 'DT'), ('frequent', 'JJ'), ('support', 'NN'), ('of',
'IN'), ('Cicero', 'NNP'), ('.', '.'), ('Caesar', 'NNP'), ('rose', 'VBD'),
('to', 'TO'), ('become', 'VB'), ('one', 'CD'), ('of', 'IN'), ('the', 'DT'),
('most', 'RBS'), ('powerful', 'JJ'), ('politicians', 'NNS'), ('in', 'IN'),
('the', 'DT'), ('Roman', 'NNP'), ('Republic', 'NNP'), ('through', 'IN'),
('a', 'DT'), ('number', 'NN'), ('of', 'IN'), ('his', 'PRP$'), ('accomplishme
nts', 'NNS'), (',', ','), ('notably', 'RB'), ('his', 'PRP$'), ('victories',
'NNS'), ('in', 'IN'), ('the', 'DT'), ('Gallic', 'NNP'), ('Wars', 'NNP'),
(',', ','), ('completed', 'VBN'), ('by', 'IN'), ('51', 'CD'), ('BC', 'NNP'),
('.', '.'), ('During', 'IN'), ('this', 'DT'), ('time', 'NN'), (',', ','),
('Caesar', 'NNP'), ('became', 'VBD'), ('the', 'DT'), ('first', 'JJ'), ('Roma
n', 'NNP'), ('general', 'NN'), ('to', 'TO'), ('cross', 'VB'), ('both', 'D
T'), ('the', 'DT'), ('English', 'NNP'), ('Channel', 'NNP'), ('and', 'CC'),
('the', 'DT'), ('Rhine', 'NNP'), ('River', 'NNP'), (',', ','), ('when', 'WR
B'), ('he', 'PRP'), ('built', 'VBD'), ('a', 'DT'), ('bridge', 'NN'), ('acros
s', 'IN'), ('the', 'DT'), ('Rhine', 'NNP'), ('and', 'CC'), ('crossed', 'VB
D'), ('the', 'DT'), ('Channel', 'NNP'), ('to', 'TO'), ('invade', 'VB'), ('Br
itain', 'NNP'), ('.', '.'), ('Caesar', 'NNP'), ("'s", 'POS'), ('wars', 'NN
S'), ('extended', 'VBD'), ('Rome', 'NNP'), ("'s", 'POS'), ('territory', 'N
N'), ('to', 'TO'), ('Britain', 'NNP'), ('and', 'CC'), ('past', 'JJ'), ('Gau
l', 'NNP'), ('.', '.'), ('These', 'DT'), ('achievements', 'NNS'), ('grante
d', 'VBD'), ('him', 'PRP'), ('unmatched', 'JJ'), ('military', 'JJ'), ('powe
r', 'NN'), ('and', 'CC'), ('threatened', 'VBD'), ('to', 'TO'), ('eclipse',
'VB'), ('the', 'DT'), ('standing', 'NN'), ('of', 'IN'), ('Pompey', 'NNP'),
(',', ','), ('who', 'WP'), ('had', 'VBD'), ('realigned', 'VBN'), ('himself',
'PRP'), ('with', 'IN'), ('the', 'DT'), ('Senate', 'NNP'), ('after', 'IN'),
('the', 'DT'), ('death', 'NN'), ('of', 'IN'), ('Crassus', 'NNP'), ('in', 'I
N'), ('53', 'CD'), ('BC', 'NNP'), ('.', '.'), ('With', 'IN'), ('the', 'DT'),
('Gallic', 'NNP'), ('Wars', 'NNP'), ('concluded', 'VBD'), (',', ','), ('th
e', 'DT'), ('Senate', 'NNP'), ('ordered', 'VBD'), ('Caesar', 'NNP'), ('to',
'TO'), ('step', 'VB'), ('down', 'RP'), ('from', 'IN'), ('his', 'PRP$'), ('mi
litary', 'JJ'), ('command', 'NN'), ('and', 'CC'), ('return', 'NN'), ('to',
'TO'), ('Rome', 'NNP'), ('.', '.'), ('Leaving', 'VBG'), ('his', 'PRP$'), ('c
ommand', 'NN'), ('in', 'IN'), ('Gaul', 'NNP'), ('meant', 'NN'), ('losing',
'VBG'), ('his', 'PRP$'), ('immunity', 'NN'), ('from', 'IN'), ('being', 'VB
G'), ('charged', 'VBN'), ('as', 'IN'), ('a', 'DT'), ('criminal', 'NN'), ('fo
r', 'IN'), ('waging', 'VBG'), ('unsanctioned', 'JJ'), ('wars', 'NNS'), ('.',
'.'), ('As', 'IN'), ('a', 'DT'), ('result', 'NN'), (',', ','), ('Caesar', 'N
NP'), ('found', 'VBD'), ('himself', 'PRP'), ('with', 'IN'), ('no', 'DT'),
('other', 'JJ'), ('options', 'NNS'), ('but', 'CC'), ('to', 'TO'), ('cross',
'VB'), ('the', 'DT'), ('Rubicon', 'NNP'), ('with', 'IN'), ('the', 'DT'), ('1
3th', 'CD'), ('Legion', 'NNP'), ('in', 'IN'), ('49', 'CD'), ('BC', 'NNP'),
(',', ','), ('leaving', 'VBG'), ('his', 'PRP$'), ('province', 'NN'), ('and',
'CC'), ('illegally', 'RB'), ('entering', 'VBG'), ('Roman', 'NNP'), ('Italy',
'NNP'), ('under', 'IN'), ('arms', 'NNS'), ('.', '.'), ('This', 'DT'), ('bega
n', 'VBD'), ('Caesar', 'NNP'), ("'s", 'POS'), ('civil', 'JJ'), ('war', 'N
N'), (',', ','), ('and', 'CC'), ('his', 'PRP$'), ('victory', 'NN'), ('in',
'IN'), ('the', 'DT'), ('war', 'NN'), ('by', 'IN'), ('45', 'CD'), ('BC', 'NN
P'), ('put', 'VBD'), ('him', 'PRP'), ('in', 'IN'), ('an', 'DT'), ('unrivale
d', 'JJ'), ('position', 'NN'), ('of', 'IN'), ('power', 'NN'), ('and', 'CC'),
('influence', 'NN'), ('.', '.')]
```

In [42]:

```python
porter = PorterStemmer()
lancaster=LancasterStemmer()
print("{0:20}{1:20}{2:20}".format("Word","Porter Stemmer","lancaster Stemmer"))
print("-" *60)
for word in words:
    print("{0:20}{1:20}{2:20}".format(word,porter.stem(word),lancaster.stem(word)))
```

```
Word                Porter Stemmer      lancaster Stemmer
------------------------------------------------------------
Gaius               gaiu                gai
Julius              juliu               juli
Caesar              caesar              caes
(                   (                   (
12                  12                  12
July                juli                july
100                 100                 100
BC                  BC                  bc
–                   –                   –
15                  15                  15
March               march               march
44                  44                  44
BC                  BC                  bc
)                   )                   )
,                   ,                   ,
known               known               known
simply              simpli              simply
```

In [43]:

```
token_words=word_tokenize(text)
stem_sentence=[]
for word in token_words:
    stem_sentence.append(porter.stem(word))
    stem_sentence.append(" ")
#print(stem_sentence)
print( "".join(stem_sentence))
```

```
gaiu juliu caesar ( 12 juli 100 BC – 15 march 44 BC ) , known simpli as juli
u caesar , wa a roman gener and statesman who play a critic role in the even
t that led to the demis of the roman republ and the rise of the roman empir
. He wa also a historian and author of latin prose . In 60 BC , caesar , cra
ssu and pompey form the first triumvir , a polit allianc that domin roman po
lit for sever year . their attempt to amass power as popular were oppos by t
he optim within the roman senat , among them cato the younger with the frequ
ent support of cicero . caesar rose to becom one of the most power politicia
n in the roman republ through a number of hi accomplish , notabl hi victori
in the gallic war , complet by 51 BC . dure thi time , caesar becam the firs
t roman gener to cross both the english channel and the rhine river , when h
e built a bridg across the rhine and cross the channel to invad britain . ca
esar 's war extend rome 's territori to britain and past gaul . these achiev
grant him unmatch militari power and threaten to eclips the stand of pompey
, who had realign himself with the senat after the death of crassu in 53 BC
. with the gallic war conclud , the senat order caesar to step down from hi
militari command and return to rome . leav hi command in gaul meant lose hi
immun from be charg as a crimin for wage unsanct war . As a result , caesar
found himself with no other option but to cross the rubicon with the 13th le
gion in 49 BC , leav hi provinc and illeg enter roman itali under arm . thi
began caesar 's civil war , and hi victori in the war by 45 BC put him in an
unriv posit of power and influenc .
```

Cosine Similarity

In [44]:

```
data_1 = "Data is the oil of the digital economy"
data_2 = "Data is a new oil"

data = [data_1, data_2]
```

Using CountVectorizer

In [45]:

```
from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer()
vector_matrix = count_vect.fit_transform(data)
print(vector_matrix)
```

```
  (0, 0)	1
  (0, 3)	1
  (0, 7)	2
  (0, 6)	1
  (0, 5)	1
  (0, 1)	1
  (0, 2)	1
  (1, 0)	1
  (1, 3)	1
  (1, 6)	1
  (1, 4)	1
```

In [46]:

```
tokens = count_vect.get_feature_names()
print(tokens)
```

```
['data', 'digital', 'economy', 'is', 'new', 'of', 'oil', 'the']
```

In [47]:

```
vocab = count_vect.vocabulary_
vocab
```

Out[47]:

```
{'data': 0,
 'is': 3,
 'the': 7,
 'oil': 6,
 'of': 5,
 'digital': 1,
 'economy': 2,
 'new': 4}
```

In [48]:

```
vec_data_1 = count_vect.transform([data_1]).toarray()
print(vec_data_1)
```

```
[[1 1 1 1 0 1 1 2]]
```

In [49]:

```
vec_data_2 = count_vect.transform([data_2]).toarray()
print(vec_data_2)
```

```
[[1 0 0 1 1 0 1 0]]
```

In [50]:

```
matrix = vector_matrix.toarray()
print(matrix)
```

```
[[1 1 1 1 0 1 1 2]
 [1 0 0 1 1 0 1 0]]
```

In [51]:

```
import pandas as pd

def create_dataframe(matrix, tokens):

    doc_names = [f'doc_{i+1}' for i, _ in enumerate(matrix)]
    df = pd.DataFrame(data=matrix, index=doc_names, columns=tokens)
    return(df)
```

In [52]:

```
create_dataframe(matrix,tokens)
```

Out[52]:

	data	digital	economy	is	new	of	oil	the
doc_1	1	1	1	1	0	1	1	2
doc_2	1	0	0	1	1	0	1	0

In [53]:

```
from sklearn.metrics.pairwise import cosine_similarity

cosine_similarity_matrix = cosine_similarity(vector_matrix)
create_dataframe(cosine_similarity_matrix,['doc_1','doc_2'])
```

Out[53]:

	doc_1	doc_2
doc_1	1.000000	0.474342
doc_2	0.474342	1.000000

In [54]:

```
print("Cosine Similarity = ",(cosine_similarity(vec_data_1,vec_data_2))[0][0])
```

```
Cosine Similarity =  0.4743416490252569
```

Using TfidfVectorizer

In [55]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer

Tfidf_vect = TfidfVectorizer()
vector_matrix = Tfidf_vect.fit_transform(data)

tokens = Tfidf_vect.get_feature_names()
create_dataframe(vector_matrix.toarray(),tokens)
```

Out[55]:

	data	digital	economy	is	new	of	oil	the
doc_1	0.243777	0.34262	0.34262	0.243777	0.000000	0.34262	0.243777	0.68524
doc_2	0.448321	0.00000	0.00000	0.448321	0.630099	0.00000	0.448321	0.00000

In [56]:

```python
cosine_similarity_matrix = cosine_similarity(vector_matrix)
create_dataframe(cosine_similarity_matrix,['doc_1','doc_2'])
```

Out[56]:

	doc_1	doc_2
doc_1	1.000000	0.327871
doc_2	0.327871	1.000000

In [57]:

```python
vec_data_1 = Tfidf_vect.transform([data_1]).toarray()
print(vec_data_1)
```

```
[[0.24377685 0.34261985 0.34261985 0.24377685 0.         0.34261985
  0.24377685 0.68523971]]
```

In [58]:

```python
vec_data_2 = Tfidf_vect.transform([data_2]).toarray()
print(vec_data_2)
```

```
[[0.44832087 0.         0.         0.44832087 0.63009934 0.
  0.44832087 0.        ]]
```

In [59]:

```python
print("Cosine Similarity = ",(cosine_similarity(vec_data_1,vec_data_2))[0][0])
```

```
Cosine Similarity =  0.3278707471841718
```

Simple Text Classifier

In [60]:

```
from nltk.corpus import names
import random
```

In [61]:

```
male_name =[(name, 'male') for name in names.words('male.txt')]
female_name = [(name, 'female') for name in names.words('female.txt')]
```

In [62]:

```
print(male_name,female_name)
```

```
[('Aamir', 'male'), ('Aaron', 'male'), ('Abbey', 'male'), ('Abbie', 'mal
e'), ('Abbot', 'male'), ('Abbott', 'male'), ('Abby', 'male'), ('Abdel', 'm
ale'), ('Abdul', 'male'), ('Abdulkarim', 'male'), ('Abdullah', 'male'),
('Abe', 'male'), ('Abel', 'male'), ('Abelard', 'male'), ('Abner', 'male'),
('Abraham', 'male'), ('Abram', 'male'), ('Ace', 'male'), ('Adair', 'mal
e'), ('Adam', 'male'), ('Adams', 'male'), ('Addie', 'male'), ('Adger', 'ma
le'), ('Aditya', 'male'), ('Adlai', 'male'), ('Adnan', 'male'), ('Adolf',
'male'), ('Adolfo', 'male'), ('Adolph', 'male'), ('Adolphe', 'male'), ('Ad
olpho', 'male'), ('Adolphus', 'male'), ('Adrian', 'male'), ('Adrick', 'mal
e'), ('Adrien', 'male'), ('Agamemnon', 'male'), ('Aguinaldo', 'male'), ('A
guste', 'male'), ('Agustin', 'male'), ('Aharon', 'male'), ('Ahmad', 'mal
e'), ('Ahmed', 'male'), ('Ahmet', 'male'), ('Ajai', 'male'), ('Ajay', 'mal
e'), ('Al', 'male'), ('Alaa', 'male'), ('Alain', 'male'), ('Alan', 'mal
e'), ('Alasdair', 'male'), ('Alastair', 'male'), ('Albatros', 'male'), ('A
lbert', 'male'), ('Alberto', 'male'), ('Albrecht', 'male'), ('Alden', 'mal
e'), ('Aldis', 'male'), ('Aldo', 'male'), ('Aldric', 'male'), ('Aldrich',
'male'), ('Aldus', 'male'), ('Aldwin', 'male'), ('Alec', 'male'), ('Alec
k', 'male'), ('Alejandro', 'male'), ('Aleks', 'male'), ('Aleksandrs', 'mal
e'), ('Alessandro', 'male'), ('Alex', 'male'), ('Alexander', 'male'), ('Al
```

In [63]:

```
labelled_name = male_name + female_name
random.shuffle(labelled_name)
```

In [64]:

```
print(labelled_name)
```

```
[('Ree', 'female'), ('Arda', 'female'), ('Remington', 'male'), ('Derrek',
'male'), ('Ahmad', 'male'), ('Ardine', 'female'), ('Irina', 'female'), ('R
afaelita', 'female'), ('Udale', 'male'), ('Engelbart', 'male'), ('Orelee',
'female'), ('Randolf', 'male'), ('Juliana', 'female'), ('Candra', 'femal
e'), ('Pierson', 'male'), ('Austin', 'female'), ('Batsheva', 'female'),
('Bert', 'male'), ('Carrol', 'female'), ('Gifford', 'male'), ('Nissa', 'fe
male'), ('Chelton', 'male'), ('Avrom', 'male'), ('Laurance', 'male'), ('Ra
monda', 'female'), ('Yance', 'male'), ('Coriss', 'female'), ('Mace', 'mal
e'), ('Giovanni', 'male'), ('Donal', 'male'), ('Aggy', 'female'), ('Dayn
a', 'female'), ('Maegan', 'female'), ('Thornie', 'male'), ('Korry', 'femal
e'), ('Daniela', 'female'), ('Maye', 'female'), ('Quintina', 'female'),
('Gilburt', 'male'), ('George', 'female'), ('Hercule', 'male'), ('Saunch
o', 'male'), ('Allina', 'female'), ('Lenna', 'female'), ('Annabal', 'femal
e'), ('Carlin', 'male'), ('Aleks', 'male'), ('Kenn', 'male'), ('Webb', 'ma
le'), ('Quintin', 'male'), ('Hugh', 'male'), ('Corabel', 'female'), ('Benn
y', 'male'), ('Prasun', 'male'), ('Bernhard', 'male'), ('Lesly', 'femal
e'), ('Evaleen', 'female'), ('Hermann', 'male'), ('Adria', 'female'), ('Au
dy', 'female'), ('Leda', 'female'), ('Harriott', 'female'), ('Emmi', 'fema
le'), ('Aleck', 'male'), ('Davina', 'female'), ('Ariella', 'female'), ('Or
```

In [65]:

```
print(len(labelled_name))
```

```
7944
```

In [66]:

```
def gender_features(word): #gives last letter of the word
    return {'last_letter':word[-1]}
```

In [67]:

```
featuresets = [(gender_features(n),gender) for (n,gender) in labelled_name]
```

In [68]:

```
featuresets
```

Out[68]:

```
[({'last_letter': 'e'}, 'female'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 'n'}, 'male'),
 ({'last_letter': 'k'}, 'male'),
 ({'last_letter': 'd'}, 'male'),
 ({'last_letter': 'e'}, 'female'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 'e'}, 'male'),
 ({'last_letter': 't'}, 'male'),
 ({'last_letter': 'e'}, 'female'),
 ({'last_letter': 'f'}, 'male'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 'n'}, 'male'),
 ({'last_letter': 'n'}, 'female'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 't'}, 'male'),
```

In [69]:

```
train_set, test_set = featuresets[500:], featuresets[:500]
```

In [70]:

```
print(len(train_set),len(test_set))
```

```
7444 500
```

In [71]:

```
import nltk
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

In [72]:

```
train_set_acc = nltk.classify.accuracy(classifier, train_set)
test_set_acc = nltk.classify.accuracy(classifier, test_set)
```

In [73]:

```
print("Accuracy on Train dataset = ", train_set_acc)
print("Accuracy on Test dataset = ", test_set_acc)
```

```
Accuracy on Train dataset =  0.7619559376679205
Accuracy on Test dataset =  0.776
```

In [74]:

```
classifier.classify(gender_features("Kavianand"))
```

Out[74]:

```
'male'
```

In [75]:

```
classifier.classify(gender_features("Kavi"))
```

Out[75]:

```
'female'
```

In [76]:

```
classifier.classify(gender_features("Kavin"))
```

Out[76]:

```
'male'
```

In [77]:

```
classifier.classify(gender_features("Rose"))
```

Out[77]:

```
'female'
```

---End of Documentation---

Submitted on 13-June-2020

Submitted by Kavianand G