```
In [2]:  #TASK 1 DEMOSTRATING COCA CORPUS
```

```
In [13]: #TASK 2 STEMMING:
         import nltk
         from nltk.stem import PorterStemmer
         Stemmerporter=PorterStemmer()
         Stemmerporter.stem('lemmatization')

Out[13]: 'lemmat'
```

```
In [14]: #TASK 2 STEMMING:
         import nltk
         from nltk.stem import PorterStemmer
         Stemmerporter=PorterStemmer()
         Stemmerporter.stem('cheerfulness')

Out[14]: 'cheer'
```

```
In [15]: import nltk
         from nltk.stem import LancasterStemmer
         stemmerLan =LancasterStemmer()
         stemmerLan.stem('happiness')

Out[15]: 'happy'
```

```
In [16]: import nltk
         from nltk.stem import RegexpStemmer
         stemmerregexp=RegexpStemmer('learn')
         stemmerregexp.stem('learning')

Out[16]: 'ing'
```

```
In [17]: import nltk
         from nltk.stem import SnowballStemmer
         SnowballStemmer.languages
         frenchstemmer=SnowballStemmer('french')
         frenchstemmer.stem('manges')

Out[17]: 'mang'
```

```
In [18]: #TASK 3: STEMMING PARAGRAPHS

         from nltk.stem import PorterStemmer

         stemmer = PorterStemmer()
         example = "Am quick brown fox jumps over a lazy dog"
         example = [stemmer.stem(token) for token in example.split(" ")]
         print (" ".join(example))

         Am quick brown fox jump over a lazi dog
```

In [19]:
```
# TASK 4: LEMMATIZER
import nltk
from nltk.corpus import wordnet as wn
from nltk.stem.wordnet import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize("cacti"))
print(lemmatizer.lemmatize("mice"))
print(lemmatizer.lemmatize("rocks"))

print(lemmatizer.lemmatize("better", pos = 'a')) # given the part-of-speech, bet

print(lemmatizer.lemmatize("Am")) # This error is fixed when the PArt of Speec

print(lemmatizer.lemmatize("am", pos = 'v'))
```

```
cactus
mouse
rock
good
Am
be
```

In [20]:
```
# TASK 5: CHINESE SEGMENTATION USING JIEBA

import jieba
seg = jieba.cut("把句子中所有的可以成词的词语都扫描出来", cut_all = True)
print(" ".join(seg))
```

```
把 句子 中所 所有 的 可以 成 词 的 词语 都 扫描 描出 描出来 出来
```

In [21]:
```python
# TASK 6: BASIC TEXT PROCESSING PIPELINE
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
import nltk
sent = "Become an expert in NLP"
words = nltk.word_tokenize(sent)
print(words)

texts = ["""The only true wisdom is in knowin' you know nothing.
Beware the barrenness of a busy life.
I decided that it was not wisdom that enabled poets to write their poetry,
but a kind of ins. or inspiration, such as you find in seers and prophets who de
count=0;
for text in texts:
    sentences = nltk.sent_tokenize(text)
    for sentence in sentences:
        words = nltk.word_tokenize(sentence)
        print(words)
        tagged = nltk.pos_tag(words)
        print(tagged)
num_words = [len(sentence.split()) for sentence in texts]
print('total word',num_words)
```

```
['Become', 'an', 'expert', 'in', 'NLP']
['The', 'only', 'true', 'wisdom', 'is', 'in', 'knowin', "'", 'you', 'know', 'no
thing', '.']
[('The', 'DT'), ('only', 'JJ'), ('true', 'JJ'), ('wisdom', 'NN'), ('is', 'VB
Z'), ('in', 'IN'), ('knowin', 'NN'), ("'", "''"), ('you', 'PRP'), ('know', 'VB
P'), ('nothing', 'NN'), ('.', '.')]
['Beware', 'the', 'barrenness', 'of', 'a', 'busy', 'life', '.']
[('Beware', 'NNP'), ('the', 'DT'), ('barrenness', 'NN'), ('of', 'IN'), ('a', 'D
T'), ('busy', 'JJ'), ('life', 'NN'), ('.', '.')]
['I', 'decided', 'that', 'it', 'was', 'not', 'wisdom', 'that', 'enabled', 'poet
s', 'to', 'write', 'their', 'poetry', ',', 'but', 'a', 'kind', 'of', 'ins',
'.']
[('I', 'PRP'), ('decided', 'VBD'), ('that', 'IN'), ('it', 'PRP'), ('was', 'VB
D'), ('not', 'RB'), ('wisdom', 'JJ'), ('that', 'IN'), ('enabled', 'VBD'), ('poe
ts', 'NNS'), ('to', 'TO'), ('write', 'VB'), ('their', 'PRP$'), ('poetry', 'N
N'), (',', ','), ('but', 'CC'), ('a', 'DT'), ('kind', 'NN'), ('of', 'IN'), ('in
s', 'NNS'), ('.', '.')]
['or', 'inspiration', ',', 'such', 'as', 'you', 'find', 'in', 'seers', 'and',
'prophets', 'who', 'deliver', 'all', 'their', 'sublime', 'messages', 'without',
'knowing', 'in', 'the', 'least', 'what', 'they', 'mean', '.']
[('or', 'CC'), ('inspiration', 'NN'), (',', ','), ('such', 'JJ'), ('as', 'IN'),
('you', 'PRP'), ('find', 'VBP'), ('in', 'IN'), ('seers', 'NNS'), ('and', 'CC'),
('prophets', 'NNS'), ('who', 'WP'), ('deliver', 'VBP'), ('all', 'DT'), ('thei
r', 'PRP$'), ('sublime', 'NN'), ('messages', 'NNS'), ('without', 'IN'), ('knowi
ng', 'VBG'), ('in', 'IN'), ('the', 'DT'), ('least', 'JJS'), ('what', 'WP'), ('t
hey', 'PRP'), ('mean', 'VBP'), ('.', '.')]
['Be', 'as', 'you', 'wish', 'to', 'seem', '.']
[('Be', 'VB'), ('as', 'IN'), ('you', 'PRP'), ('wish', 'VBP'), ('to', 'TO'), ('s
eem', 'VB'), ('.', '.')]
['Wonder', 'is', 'the', 'beginning', 'of', 'wisdom', '.']
[('Wonder', 'NNP'), ('is', 'VBZ'), ('the', 'DT'), ('beginning', 'NN'), ('of',
'IN'), ('wisdom', 'NN'), ('.', '.')]
```

```
['Be', 'kind', ',', 'for', 'everyone', 'you', 'meet', 'is', 'fighting', 'a', 'h
ard', 'battle', '.']
[('Be', 'NNP'), ('kind', 'NN'), (',', ','), ('for', 'IN'), ('everyone', 'NN'),
('you', 'PRP'), ('meet', 'VBP'), ('is', 'VBZ'), ('fighting', 'VBG'), ('a', 'D
T'), ('hard', 'JJ'), ('battle', 'NN'), ('.', '.')]
['Our', 'prayers', 'should', 'be', 'for', 'blessings', 'in', 'general', ',', 'f
or', 'God', 'knows', 'best', 'what', 'is', 'good', 'for', 'us', '.']
[('Our', 'PRP$'), ('prayers', 'NNS'), ('should', 'MD'), ('be', 'VB'), ('for',
'IN'), ('blessings', 'NNS'), ('in', 'IN'), ('general', 'JJ'), (',', ','), ('fo
r', 'IN'), ('God', 'NNP'), ('knows', 'VBZ'), ('best', 'JJS'), ('what', 'WP'),
('is', 'VBZ'), ('good', 'JJ'), ('for', 'IN'), ('us', 'PRP'), ('.', '.')]
total word [100]

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

In [ ]:

In [ ]:

In [3]:
```
#stopword
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords.words('english')
```

```
 'further',
 'then',
 'once',
 'here',
 'there',
 'when',
 'where',
 'why',
 'how',
 'all',
 'any',
 'both',
 'each',
 'few',
 'more',
 'most',
 'other',

 'some',
 'such',
```

In [5]:
```
#CMU wordlist
import nltk
nltk.download('cmudict')
import nltk
entries=nltk.corpus.cmudict.entries()
len(entries)
```

```
[nltk_data] Downloading package cmudict to
[nltk_data]   C:\Users\kalyani\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\cmudict.zip.
```

Out[5]:
```
133737
```

In [6]:
```
from nltk.corpus import wordnet as wn
wn.synsets('motorcar')
```

Out[6]:
```
[Synset('car.n.01')]
```

In [7]:
```
wn.synset('car.n.01').lemma_names()
```

Out[7]:
```
['car', 'auto', 'automobile', 'machine', 'motorcar']
```

In [16]:
```
#TASK CLASSIFIER
def gender_features(word):
    return {'last_letter':word[-1]}
```

```
In [17]: gender_features('obama')

Out[17]: {'last_letter': 'a'}
```

```
In [12]: import nltk
         nltk.download('names')
         from nltk.corpus import names
         labeled_names = ([(name, 'male') for name in names.words('male.txt')]+
         [(name, 'female') for name in names.words('female.txt')])

         [nltk_data] Downloading package names to
         [nltk_data]   C:\Users\kalyani\AppData\Roaming\nltk_data...
         [nltk_data]   Unzipping corpora\names.zip.
```

```
In [14]: import random
         random.shuffle(labeled_names)
```

```
In [15]: featuresets=[(gender_features(n),gender) for (n,gender) in labeled_names]
```

```
In [18]: train_set,test_test=featuresets[500:],featuresets[:500]
```

```
In [20]: import nltk
         classifier=nltk.NaiveBayesClassifier.train(train_set)
```

```
In [21]: classifier.classify(gender_features('David'))

Out[21]: 'male'
```

```
In [22]: classifier.classify(gender_features('Michelle'))

Out[22]: 'female'
```

```
In [23]: classifier.classify(gender_features('obama'))

Out[23]: 'female'
```

```
In [26]: classifier.classify(gender_features('Alex'))

Out[26]: 'female'
```

```
In [25]: print(nltk.classify.accuracy(classifier,test_test))

         0.772
```

```
In [27]: #Task 3 Vectoriser and cosine similarity   vectoriser used for word to number
         from sklearn.feature_extraction.text import CountVectorizer
         #from sklearn.feature_extraction.text import TfidVectorizer
```

In [28]:
```
vect=CountVectorizer(binary=True)
corpus = ["Tessaract is good optical character recognition engine  ", "optical c
vect.fit(corpus)
```

Out[28]:
```
CountVectorizer(analyzer='word', binary=True, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

In [29]:
```
vocab=vect.vocabulary_
```

In [30]:
```
for key in sorted(vocab.keys()):
    print("{}:{}".format(key,vocab[key]))
```

```
character:0
engine:1
good:2
is:3
optical:4
recognition:5
significant:6
tessaract:7
```

In [31]:
```
print(vect.transform(["this is a good optical illusion"]).toarray())
```

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

In [32]:
```
print(vect.transform(corpus).toarray())
```

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

In [34]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

In [36]:
```
#simalrity between two sentence from given corpus
similarity = cosine_similarity(vect.transform(["Google Cloud Vision is a charact
```

In [37]:
```
print(similarity)
```

```
[[0.89442719]]
```

In [ ]:

In [1]:
```
import nltk
```

In [2]:
```
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xm
l (https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)
```

Out[2]:
```
True
```

In [3]:
```
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

In [7]:
```
# Task 2: Imort Brown Corpus and Accessing Data
from nltk.corpus import brown

# categories in brown corpora
#brown.categories()
#brown.words(categories='adventure')[:20]
brown.words(categories='fiction')[:20]
```

Out[7]:
```
['Thirty-three',
 'Scotty',
 'did',
 'not',
 'go',
 'back',
 'to',
 'school',
 '.',
 'His',
 'parents',
 'talked',
 'seriously',
 'and',
 'lengthily',
 'to',
 'their',
 'own',
 'doctor',
 'and']
```

In [11]:
```
# Task 3: Import Inaugral Corpus and access data
# includes every president's inaugral address from 1789 to 2009
from nltk.corpus import inaugural
inaugural.fileids()
```

Out[11]:
```
['1789-Washington.txt',
 '1793-Washington.txt',
 '1797-Adams.txt',
 '1801-Jefferson.txt',
 '1805-Jefferson.txt',
 '1809-Madison.txt',
 '1813-Madison.txt',
 '1817-Monroe.txt',
 '1821-Monroe.txt',
 '1825-Adams.txt',
 '1829-Jackson.txt',
 '1833-Jackson.txt',
 '1837-VanBuren.txt',
 '1841-Harrison.txt',
 '1845-Polk.txt',
 '1849-Taylor.txt',
 '1853-Pierce.txt',
 '1857-Buchanan.txt',
 '1861-Lincoln.txt',
 '1865-Lincoln.txt',
 '1869-Grant.txt',
 '1873-Grant.txt',
 '1877-Hayes.txt',
 '1881-Garfield.txt',
 '1885-Cleveland.txt',
 '1889-Harrison.txt',
 '1893-Cleveland.txt',
 '1897-McKinley.txt',
 '1901-McKinley.txt',
 '1905-Roosevelt.txt',
 '1909-Taft.txt',
 '1913-Wilson.txt',
 '1917-Wilson.txt',
 '1921-Harding.txt',
 '1925-Coolidge.txt',
 '1929-Hoover.txt',
 '1933-Roosevelt.txt',
 '1937-Roosevelt.txt',
 '1941-Roosevelt.txt',
 '1945-Roosevelt.txt',
 '1949-Truman.txt',
 '1953-Eisenhower.txt',
 '1957-Eisenhower.txt',
 '1961-Kennedy.txt',
 '1965-Johnson.txt',
 '1969-Nixon.txt',
 '1973-Nixon.txt',
 '1977-Carter.txt',
 '1981-Reagan.txt',
 '1985-Reagan.txt',
 '1989-Bush.txt',
 '1993-Clinton.txt',
 '1997-Clinton.txt',
 '2001-Bush.txt',
 '2005-Bush.txt',
 '2009-Obama.txt',
 '2013-Obama.txt',
 '2017-Trump.txt']
```

In [15]:
```
inaugural.words(fileids = '2017-Trump.txt')[:50]
```

Out[15]:
```
['Chief',
 'Justice',
 'Roberts',
 ',',
 'President',
 'Carter',
 ',',
 'President',
 'Clinton',
 ',',
 'President',
 'Bush',
 ',',
 'President',
 'Obama',
 ',',
 'fellow',
 'Americans',
 ',',
 'and',
 'people',
 'of',
 'the',
 'world',
 ':',
 'Thank',
 'you',
 '.',
 'We',
 ',',
 'the',
 'citizens',
 'of',
 'America',
 ',',
 'are',
 'now',
 'joined',
 'in',
 'a',
 'great',
 'national',
 'effort',
 'to',
 'rebuild',
 'our',
 'country',
 'and',
 'restore',
 'its']
```

```
In [14]: inaugural.words(fileids = '1861-Lincoln.txt')[:50]

Out[14]: ['Fellow',
          '-',
          'Citizens',
          'of',
          'the',
          'United',
          'States',
          ':',
          'In',
          'compliance',
          'with',
          'a',
          'custom',
          'as',
          'old',
          'as',
          'the',
          'Government',
          'itself',
          ',',
          'I',
          'appear',
          'before',
          'you',
          'to',
          'address',
          'you',
          'briefly',
          'and',
          'to',
          'take',
          'in',
          'your',
          'presence',
          'the',
          'oath',
          'prescribed',
          'by',
          'the',
          'Constitution',
          'of',
          'the',
          'United',
          'States',
          'to',
          'be',
          'taken',
          'by',
          'the',
          'President']
```

In [16]:
```
inaugural.words(fileids = '2009-Obama.txt')[:50]
print(inaugural.words(fileids = '2009-Obama.txt'))
```

Out[16]:
```
['My',
 'fellow',
 'citizens',
 ':',
 'I',
 'stand',
 'here',
 'today',
 'humbled',
 'by',
 'the',
 'task',
 'before',
 'us',
 ',',
 'grateful',
 'for',
 'the',
 'trust',
 'you',
 'have',
 'bestowed',
 ',',
 'mindful',
 'of',
 'the',
 'sacrifices',
 'borne',
 'by',
 'our',
 'ancestors',
 '.',
 'I',
 'thank',
 'President',
 'Bush',
 'for',
 'his',
 'service',
 'to',
 'our',
 'nation',
 ',',
 'as',
 'well',
 'as',
 'the',
 'generosity',
 'and',
 'cooperation']
```

In [17]:
```
#Task 4: Importing WEBTEXT CORPUS and Access Data
from nltk.corpus import webtext
webtext.fileids()

for fileid in webtext.fileids():
    print(fileid, webtext.raw(fileid)[:])
```

```
firefox.txt Cookie Manager: "Don't allow sites that set removed cookies to se
t future cookies" should stay checked
When in full screen mode
Pressing Ctrl-N should open a new browser when only download dialog is left o
pen
add icons to context menu
So called "tab bar" should be made a proper toolbar or given the ability coll
apse / expand.
[XUL] Implement Cocoa-style toolbar customization.
#ifdefs for MOZ_PHOENIX
customize dialog's toolbar has small icons when small icons is not checked
nightly builds and tinderboxen for Phoenix
finish tearing prefs UI to pieces and then make it not suck
"mozbrowser" script doesn't start correct binary
Need bookmark groups icon
Dropping at top of palette box horks things
keyboard shortcut for Increase Text Size is broken
default phoenix bookmarks
[cust] need a toolbar spacer and spring spacer for customize
```

In [18]:
```
# Task 5: Frequency Distribution of words in a text
text1 = '''1962 Tour de France was the 49th edition of the Tour de France, one o
fd = nltk.FreqDist(text1.split())
```

In [21]:
```
fd
```

Out[21]:
```
FreqDist({'the': 6, 'of': 5, 'Tour': 4, 'de': 3, 'was': 3, 'in': 3, 'and': 3,
'stages,': 2, 'on': 2, 'his': 2, ...})
```

In [23]:
```
# Task 6. Conditional Frequency Distribution of words in a text
# tells us how many 2 letter words or 3 letter words
from nltk.probability import ConditionalFreqDist
cfd = ConditionalFreqDist((len(word), word) for word in text1.split())
cfd[3]
```

Out[23]:
```
FreqDist({'the': 6, 'was': 3, 'and': 3, 'his': 2, 'one': 1, 'The': 1, 'mi)': 1,
'two': 1, 'des': 1, 'won': 1, ...})
```

In [24]:
```
cfd[6]
```

Out[24]:
```
FreqDist({'France': 1, 'Tours.': 1, '(2,656': 1, 'stages': 1, 'years,': 1, 'tea
ms.': 1, 'placed': 1, 'third,': 1, 'behind': 1})
```

In [26]:

```
['My', 'fellow', 'citizens', ':', 'I', 'stand', 'here', ...]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-cf4f9f33f6c4> in <module>
----> 1 print(inaugural.words(fileids = '2009-Obama.txt'))[0]

TypeError: 'NoneType' object is not subscriptable
```

In [ ]: