In [3]:

```
#stopword
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords.words('english')
```

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

In [5]:

```
'''CMUdict is a versioned python wrapper package for The CMU Pronouncing Dictiona
The main purpose is to expose the data with little or no assumption on how it is
#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\Roshan\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\cmudict.zip.
```

Out[5]:

```
133737
```

In [1]:

```
from nltk.corpus import wordnet as wn
'''Synset is a special kind of a simple interface that is present in NLTK to lool
Synset instances are the groupings of synonymous words that express the same conc
Some of the words have only one Synset and some have several.'''
wn.synsets('motorcar')
from nltk.corpus import wordnet
syn = wordnet.synsets('hello')[0]

print ("Synset name :  ", syn.name())

# Defining the word
print ("\nSynset meaning : ", syn.definition())

# list of phrases that use the word in context
print ("\nSynset example : ", syn.examples())
```

```
Synset name :   hello.n.01

Synset meaning :  an expression of greeting

Synset example :  ['every morning they exchanged polite hellos']
```

In [7]:

```
wn.synset('car.n.01').lemma_names()
```

Out[7]:

```
['car', 'auto', 'automobile', 'machine', 'motorcar']
```

In [16]:

```
#TASK CLASSIFIER
'''this is just for understanding for some specific example it may get wrong resu
we saw that male and female names have some distinctive characteristics.
Names ending in a, e and i are likely to be female,
while names ending in k, o, r, s and t are likely to be male.
Let's build a classifier to model these differences more precisely.'''
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\Roshan\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 cl
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 characte
```

In [37]:

```
print(similarity)
```

```
[[0.89442719]]
```

In [ ]: