In [2]:

```
import nltk
from nltk.corpus import stopwords
# list of words with no meaning
stopwords.words('english') #at last we remove the. But for sequencial we do not discard
```

Out[2]:

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
 'do',
 'does',
 'did',
 'doing',
 'a',
 'an',
 'the',
 'and',
 'but',
 'if',
 'or',
 'because',
 'as',
 'until',
 'while',
 'of',
 'at',
 'by',
 'for',
 'with',
 'about',
```

```
'against',
'between',
'into',
'through',
'during',
'before',
'after',
'above',
'below',
'to',
'from',
'up',
'down',
'in',
'out',
'on',
'off',
'over',
'under',
'again',
'further',
'then',
'once',
'here',
'there',
'when',
'where',
'why',
'how',
'all',
'any',
'both',
'each',
'few',
'more',
'most',
'other',
'some',
'such',
'no',
'nor',
'not',
'only',
'own',
'same',
'so',
'than',
'too',
'very',
's',
't',
'can',
'will',
'just',
'don',
"don't",
'should',
"should've",
'now',
'd',
'll',
'm',
'o',
're',
've',
'y',
'ain',
'aren',
"aren't",
'couldn',
"couldn't",
'didn',
"didn't",
'doesn',
"doesn't",
'hadn',
"hadn't",
'hasn',
"hasn't",
'haven',
"haven't",
'isn',
"isn't",
```

```
'ma',
'mightn',
"mightn't",
'mustn',
"mustn't",
'needn',
"needn't",
'shan',
"shan't",
'shouldn',
"shouldn't",
'wasn',
"wasn't",
'weren',
"weren't",
'won',
"won't",
'wouldn',
"wouldn't"]
```

In [3]:

```
from nltk.corpus import cmudict
entries=nltk.corpus.cmudict.entries()
len(entries)
for entry in entries[10000:10025]:
    print(entry)
```

```
('belford', ['B', 'EH1', 'L', 'F', 'ER0', 'D'])
('belfry', ['B', 'EH1', 'L', 'F', 'R', 'IY0'])
('belgacom', ['B', 'EH1', 'L', 'G', 'AH0', 'K', 'AA0', 'M'])
('belgacom', ['B', 'EH1', 'L', 'JH', 'AH0', 'K', 'AA0', 'M'])
('belgard', ['B', 'EH0', 'L', 'G', 'AA1', 'R', 'D'])
('belgarde', ['B', 'EH0', 'L', 'G', 'AA1', 'R', 'D', 'IY0'])
('belge', ['B', 'EH1', 'L', 'JH', 'IY0'])
('belger', ['B', 'EH1', 'L', 'G', 'ER0'])
('belgian', ['B', 'EH1', 'L', 'JH', 'AH0', 'N'])
('belgians', ['B', 'EH1', 'L', 'JH', 'AH0', 'N', 'Z'])
('belgique', ['B', 'EH0', 'L', 'ZH', 'IY1', 'K'])
("belgique's", ['B', 'EH0', 'L', 'JH', 'IY1', 'K', 'S'])
('belgium', ['B', 'EH1', 'L', 'JH', 'AH0', 'M'])
("belgium's", ['B', 'EH1', 'L', 'JH', 'AH0', 'M', 'Z'])
('belgo', ['B', 'EH1', 'L', 'G', 'OW2'])
('belgrade', ['B', 'EH1', 'L', 'G', 'R', 'EY0', 'D'])
('belgrade', ['B', 'EH1', 'L', 'G', 'R', 'AA2', 'D'])
("belgrade's", ['B', 'EH1', 'L', 'G', 'R', 'EY0', 'D', 'Z'])
("belgrade's", ['B', 'EH1', 'L', 'G', 'R', 'AA2', 'D', 'Z'])
('belgrave', ['B', 'EH1', 'L', 'G', 'R', 'EY2', 'V'])
('beli', ['B', 'EH1', 'L', 'IY0'])
('belich', ['B', 'EH1', 'L', 'IH0', 'K'])
('belie', ['B', 'IH0', 'L', 'AY1'])
('belied', ['B', 'IH0', 'L', 'AY1', 'D'])
('belief', ['B', 'IH0', 'L', 'IY1', 'F'])
```

In [5]:

```
from nltk.corpus import wordnet as wn
wn.synsets('motorcar')
wn.synset('car.n.01')
```

Out[5]:

```
Synset('car.n.01')
```

In [6]:

```
# STEMMING
from  nltk.stem import PorterStemmer
from  nltk.stem import LancasterStemmer
stemmerporter=PorterStemmer()
stemmerporter1=LancasterStemmer()
```

In [15]:

```
stemmerporter.stem('happiness')
```

Out[15]:

```
'happi'
```

In [12]:

```
stemmerporter.stem('Thinking')
```

Out[12]:

```
'think'
```

In [13]:
```
stemmerporter.stem('Laughing')
```

Out[13]: 'laugh'

In [14]:
```
stemmerporter1.stem('happiness')
```

Out[14]: 'happy'

In [20]:
```
from nltk.stem import RegexpStemmer
stemmerregexp=RegexpStemmer('learn')
stemmerregexp.stem('learning')
```

Out[20]: 'ing'

In [21]:
```
stemmerregexp1=RegexpStemmer('ing')
stemmerregexp1.stem('singing')
```

Out[21]: 's'

In [9]:
```
from nltk.stem import SnowballStemmer #supports internation languages
SnowballStemmer.languages
frenchstemmer=SnowballStemmer('french')
frenchstemmer.stem('manges')
```

Out[9]: 'mang'

In [10]:
```
sent="Become an expert in nlp"
words=nltk.word_tokenize(sent)
print(words)
```

```
['Become', 'an', 'expert', 'in', 'nlp']
```

In [ ]:
```
texts="Over the years, advancements in the work have been incorporated towards professor relationships and data c
for text in texts:
    sentences=nltk.sent_tokenization(text) #list of sentences
    for sentance in sentances:
        words=nltk.word_tokenize(sentance)
        print(words)
        tagged=nltk.pos_tag(words)
        print(tagged)
```