In [8]:
```
#task 2: tweettokenizer
import nltk
from nltk.tokenize import TweetTokenizer
text='NLP Class Session is Amazing: D #superfun'
twtkn=TweetTokenizer()
twtkn.tokenize(text)
```

Out[8]:
```
['NLP', 'Class', 'Session', 'is', 'Amazing', ':', 'D', '#superfun']
```

In [7]:
```
#Task 1 - Lexicons
#1. Stopwords:
from nltk.corpus import stopwords
stopwords.words('english')
```

Out[7]:
```
['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 [9]:

```
#2 CMU Wordlist
import nltk
entries=nltk.corpus.cmudict.entries()
len(entries)
```

Out[9]:

```
133737
```

In [10]:
```
print(entries[:100])
```

```
[('a', ['AH0']), ('a.', ['EY1']), ('a', ['EY1']), ('a42128', ['EY1', 'F', 'AO1',
'R', 'T', 'UW1', 'W', 'AH1', 'N', 'T', 'UW1', 'EY1', 'T']), ('aaa', ['T', 'R', 'IH
2', 'P', 'AH0', 'L', 'EY1']), ('aaberg', ['AA1', 'B', 'ER0', 'G']), ('aachen', ['AA
1', 'K', 'AH0', 'N']), ('aachener', ['AA1', 'K', 'AH0', 'N', 'ER0']), ('aaker', ['AA
1', 'K', 'ER0']), ('aalseth', ['AA1', 'L', 'S', 'EH0', 'TH']), ('aamodt', ['AA1',
'M', 'AH0', 'T']), ('aancor', ['AA1', 'N', 'K', 'AO2', 'R']), ('aardema', ['AA0',
'R', 'D', 'EH1', 'M', 'AH0']), ('aardvark', ['AA1', 'R', 'D', 'V', 'AA2', 'R',
'K']), ('aaron', ['EH1', 'R', 'AH0', 'N']), ("aaron's", ['EH1', 'R', 'AH0', 'N',
'Z']), ('aarons', ['EH1', 'R', 'AH0', 'N', 'Z']), ('aaronson', ['EH1', 'R', 'AH0',
'N', 'S', 'AH0', 'N']), ('aaronson', ['AA1', 'R', 'AH0', 'N', 'S', 'AH0', 'N']), ("a
aronson's", ['EH1', 'R', 'AH0', 'N', 'S', 'AH0', 'N', 'Z']), ("aaronson's", ['AA1',
'R', 'AH0', 'N', 'S', 'AH0', 'N', 'Z']), ('aarti', ['AA1', 'R', 'T', 'IY2']), ('aas
e', ['AA1', 'S']), ('aasen', ['AA1', 'S', 'AH0', 'N']), ('ab', ['AE1', 'B']), ('ab',
['EY1', 'B', 'IY1']), ('ababa', ['AH0', 'B', 'AA1', 'B', 'AH0']), ('ababa', ['AA1',
'B', 'AH0', 'B', 'AH0']), ('abacha', ['AE1', 'B', 'AH0', 'K', 'AH0']), ('aback', ['A
H0', 'B', 'AE1', 'K']), ('abaco', ['AE1', 'B', 'AH0', 'K', 'OW2']), ('abacus', ['AE
1', 'B', 'AH0', 'K', 'AH0', 'S']), ('abad', ['AH0', 'B', 'AA1', 'D']), ('abadaka',
['AH0', 'B', 'AE1', 'D', 'AH0', 'K', 'AH0']), ('abadi', ['AH0', 'B', 'AE1', 'D', 'IY
0']), ('abadie', ['AH0', 'B', 'AE1', 'D', 'IY0']), ('abair', ['AH0', 'B', 'EH1',
'R']), ('abalkin', ['AH0', 'B', 'AA1', 'L', 'K', 'IH0', 'N']), ('abalone', ['AE2',
'B', 'AH0', 'L', 'OW1', 'N', 'IY0']), ('abalos', ['AA0', 'B', 'AA1', 'L', 'OW0',
'Z']), ('abandon', ['AH0', 'B', 'AE1', 'N', 'D', 'AH0', 'N']), ('abandoned', ['AH0',
'B', 'AE1', 'N', 'D', 'AH0', 'N', 'D']), ('abandoning', ['AH0', 'B', 'AE1', 'N',
'D', 'AH0', 'N', 'IH0', 'NG']), ('abandonment', ['AH0', 'B', 'AE1', 'N', 'D', 'AH0',
'N', 'M', 'AH0', 'N', 'T']), ('abandonments', ['AH0', 'B', 'AE1', 'N', 'D', 'AH0',
'N', 'M', 'AH0', 'N', 'T', 'S']), ('abandons', ['AH0', 'B', 'AE1', 'N', 'D', 'AH0',
'N', 'Z']), ('abanto', ['AH0', 'B', 'AE1', 'N', 'T', 'OW0']), ('abarca', ['AH0',
'B', 'AA1', 'R', 'K', 'AH0']), ('abare', ['AA0', 'B', 'AA1', 'R', 'IY0']), ('abasca
l', ['AE1', 'B', 'AH0', 'S', 'K', 'AH0', 'L']), ('abash', ['AH0', 'B', 'AE1', 'S
H']), ('abashed', ['AH0', 'B', 'AE1', 'SH', 'T']), ('abate', ['AH0', 'B', 'EY1',
'T']), ('abated', ['AH0', 'B', 'EY1', 'T', 'IH0', 'D']), ('abatement', ['AH0', 'B',
'EY1', 'T', 'M', 'AH0', 'N', 'T']), ('abatements', ['AH0', 'B', 'EY1', 'T', 'M', 'AH
0', 'N', 'T', 'S']), ('abates', ['AH0', 'B', 'EY1', 'T', 'S']), ('abating', ['AH0',
'B', 'EY1', 'T', 'IH0', 'NG']), ('abba', ['AE1', 'B', 'AH0']), ('abbado', ['AH0',
'B', 'AA1', 'D', 'OW0']), ('abbas', ['AH0', 'B', 'AA1', 'S']), ('abbasi', ['AA0',
'B', 'AA1', 'S', 'IY0']), ('abbate', ['AA1', 'B', 'EY0', 'T']), ('abbatiello', ['AA
0', 'B', 'AA0', 'T', 'IY0', 'EH1', 'L', 'OW0']), ('abbe', ['AE1', 'B', 'IY0']), ('ab
be', ['AE0', 'B', 'EY1']), ('abbenhaus', ['AE1', 'B', 'AH0', 'N', 'HH', 'AW2',
'S']), ('abbett', ['AH0', 'B', 'EH1', 'T']), ('abbeville', ['AE1', 'B', 'V', 'IH0',
'L']), ('abbey', ['AE1', 'B', 'IY0']), ("abbey's", ['AE1', 'B', 'IY0', 'Z']), ('abbi
e', ['AE1', 'B', 'IY0']), ('abbitt', ['AE1', 'B', 'IH0', 'T']), ('abbot', ['AE1',
'B', 'AH0', 'T']), ('abbotstown', ['AE1', 'B', 'AH0', 'T', 'S', 'T', 'AW1', 'N']),
('abbott', ['AE1', 'B', 'AH0', 'T']), ("abbott's", ['AE1', 'B', 'AH0', 'T', 'S']),
('abbottstown', ['AE1', 'B', 'AH0', 'T', 'S', 'T', 'AW1', 'N']), ('abboud', ['AH0',
'B', 'UW1', 'D']), ('abboud', ['AH0', 'B', 'AW1', 'D']), ('abbreviate', ['AH0', 'B',
'R', 'IY1', 'V', 'IY0', 'EY2', 'T']), ('abbreviated', ['AH0', 'B', 'R', 'IY1', 'V',
'IY0', 'EY2', 'T', 'AH0', 'D']), ('abbreviated', ['AH0', 'B', 'R', 'IY1', 'V', 'IY
0', 'EY2', 'T', 'IH0', 'D']), ('abbreviates', ['AH0', 'B', 'R', 'IY1', 'V', 'IY0',
'EY2', 'T', 'S']), ('abbreviating', ['AH0', 'B', 'R', 'IY1', 'V', 'IY0', 'EY2', 'T',
'IH0', 'NG']), ('abbreviation', ['AH0', 'B', 'R', 'IY2', 'V', 'IY0', 'EY1', 'SH', 'A
H0', 'N']), ('abbreviations', ['AH0', 'B', 'R', 'IY2', 'V', 'IY0', 'EY1', 'SH', 'AH
0', 'N', 'Z']), ('abbruzzese', ['AA0', 'B', 'R', 'UW0', 'T', 'S', 'EY1', 'Z', 'IY
0']), ('abbs', ['AE1', 'B', 'Z']), ('abby', ['AE1', 'B', 'IY0']), ('abco', ['AE1',
'B', 'K', 'OW0']), ('abcotek', ['AE1', 'B', 'K', 'OW0', 'T', 'EH2', 'K']), ('abdall
a', ['AE2', 'B', 'D', 'AE1', 'L', 'AH0']), ('abdallah', ['AE2', 'B', 'D', 'AE1',
'L', 'AH0']), ('abdel', ['AE1', 'B', 'D', 'EH2', 'L']), ('abdella', ['AE2', 'B',
'D', 'EH1', 'L', 'AH0']), ('abdicate', ['AE1', 'B', 'D', 'AH0', 'K', 'EY2', 'T']),
('abdicated', ['AE1', 'B', 'D', 'AH0', 'K', 'EY2', 'T', 'AH0', 'D']), ('abdicates',
['AE1', 'B', 'D', 'AH0', 'K', 'EY2', 'T', 'S']), ('abdicating', ['AE1', 'B', 'D', 'I
H0', 'K', 'EY2', 'T', 'IH0', 'NG'])]
```

In [12]:
```
#3 Wordnet
from nltk.corpus import wordnet as wn
wn.synsets('motorcar')
```

```
Out[12]: [Synset('car.n.01')]
```

```
In [13]: wn.synset('car.n.01').lemma_names()
```

```
Out[13]: ['car', 'auto', 'automobile', 'machine', 'motorcar']
```

```
In [14]: wn.synsets('good')
```

```
Out[14]: [Synset('good.n.01'),
          Synset('good.n.02'),
          Synset('good.n.03'),
          Synset('commodity.n.01'),
          Synset('good.a.01'),
          Synset('full.s.06'),
          Synset('good.a.03'),
          Synset('estimable.s.02'),
          Synset('beneficial.s.01'),
          Synset('good.s.06'),
          Synset('good.s.07'),
          Synset('adept.s.01'),
          Synset('good.s.09'),
          Synset('dear.s.02'),
          Synset('dependable.s.04'),
          Synset('good.s.12'),
          Synset('good.s.13'),
          Synset('effective.s.04'),
          Synset('good.s.15'),
          Synset('good.s.16'),
          Synset('good.s.17'),
          Synset('good.s.18'),
          Synset('good.s.19'),
          Synset('good.s.20'),
          Synset('good.s.21'),
          Synset('well.r.01'),
          Synset('thoroughly.r.02')]
```

```
In [15]: wn.synset('dependable.s.04').lemma_names()
```

```
Out[15]: ['dependable', 'good', 'safe', 'secure']
```

```
In [0]:
```