In [1]:
```
import nltk
from nltk.corpus import stopwords
stopwords.words('english')
```

```
 '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 [2]:
```
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 [3]:
```
from nltk.corpus import wordnet as wn
wn.synsets('motocar')
wn.synset('car.n.01').lemma_names()
```

Out[3]:
```
['car', 'auto', 'automobile', 'machine', 'motorcar']
```

In [4]:
```
from nltk.corpus import wordnet as wn
wn.synsets('good')
```

Out[4]:
```
[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 [5]:
```
from nltk.stem import PorterStemmer
stemmerporter=PorterStemmer()
stemmerporter.stem('happiness')
```

Out[5]:
```
'happi'
```

In [6]:
```
from nltk.stem import PorterStemmer
stemmerporter=PorterStemmer()
stemmerporter.stem('happier')
```

Out[6]:
```
'happier'
```

In [7]:
```
from nltk.stem import PorterStemmer
stemmerporter=PorterStemmer()
stemmerporter.stem('unhappy')
```

Out[7]:
```
'unhappi'
```

In [8]:
```
from nltk.stem import PorterStemmer
stemmerporter=PorterStemmer()
stemmerporter.stem('backfoot')
```

Out[8]:
```
'backfoot'
```

In [9]:
```
from nltk.stem import LancasterStemmer
stemmerporter=LancasterStemmer()
stemmerporter.stem('happiness')
```

Out[9]:
```
'happy'
```

In [10]:
```
from nltk.stem import RegexpStemmer
stemmerregexp=RegexpStemmer('sing')
stemmerregexp.stem('singing')
```

Out[10]:
```
'ing'
```

In [12]:
```
from nltk.stem import SnowballStemmer
SnowballStemmer.languages
frenchstemmer=Snowballstemmer('french')
frenchstemmer.stem('manges')
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[12], line 3
      1 from nltk.stem import SnowballStemmer
      2 SnowballStemmer.languages
----> 3 frenchstemmer=Snowballstemmer('french')
      4 frenchstemmer.stem('manges')

NameError: name 'Snowballstemmer' is not defined
```

In [13]:
```
from nltk.stem import SnowballStemmer
SnowballStemmer.languages
frenchstemmer=SnowbalStemmer('french')
frenchstemmer.stem('manges')
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[13], line 3
      1 from nltk.stem import SnowballStemmer
      2 SnowballStemmer.languages
----> 3 frenchstemmer=SnowbalStemmer('french')
      4 frenchstemmer.stem('manges')

NameError: name 'SnowbalStemmer' is not defined
```

In [14]:
```
from nltk.stem import SnowballStemmer
SnowballStemmer.languages
frenchstemmer=SnowballStemmer('french')
frenchstemmer.stem('manges')
```

Out[14]:
```
'mang'
```

In [15]:
```
sent="Become an expert in NLP"
words=nltk.word_tokenize(sent)
print(words)
```

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

In [17]:
```
for text in texts:
    sentences=nltk.sent_tokeninze(text)
    for sentence in sentences:
        words=nltk.word_tokenize(sentences)
        print(words)
        tagged=nltk.pos_tag(words)
        print(tagged)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[17], line 1
----> 1 for text in texts:
      2     sentences=nltk.sent_tokeninze(text)
      3     for sentence in sentences:

NameError: name 'texts' is not defined
```

In [20]:
```
for text in Travel by freighter, the average cost of a voyage is just about

There is an additional charge of about $262.00 for deviation insurance and
    sentences=nltk.sent_tokeninze(text)
    for sentence in sentences:
        words=nltk.word_tokenize(sentences)
        print(words)
        tagged=nltk.pos_tag(words)
        print(tagged)
```

```
  Cell In[20], line 1
    for text in Travel by freighter, the average cost of a voyage is just
about $100.00 US per day, for a single person traveling in a single cabin.
It is always more expensive for a single to book a double cabin and always
cheaper per person for double occupancy of a double cabin.
                       ^
SyntaxError: invalid syntax
```

In [21]:
```
for text in texts:'Travel by freighter, the average cost of a voyage is jus
    sentences=nltk.sent_tokeninze(text)
    for sentence in sentences:
        words=nltk.word_tokenize(sentences)
        print(words)
        tagged=nltk.pos_tag(words)
        print(tagged)
```

```
  Cell In[21], line 2
    sentences=nltk.sent_tokeninze(text)
    ^
IndentationError: unexpected indent
```

In [22]:
```
for text in texts:
    sentences=nltk.sent_tokeninze(There is an additional charge of about $2
    for sentence in sentences:
        words=nltk.word_tokenize(sentences)
        print(words)
        tagged=nltk.pos_tag(words)
        print(tagged)
```

```
  Cell In[22], line 2
    sentences=nltk.sent_tokeninze(There is an additional charge of about
$262.00 for deviation insurance and a $12.50 customs charge per person dep
arting or entering the country. Keep in mind that more than one owner/char
ter may have vessels on a given route. The fare charged by different owner
s on the same route can vary considerably. Shop around.)
                                                      ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

In [7]:
```
import nltk
```

In [8]:
```
texts="A mother the epitome of unconditional love and unwavering strength h
sentences=nltk.sent_tokenize(texts)
print(sentences)
for text in texts:
    #sentences=nltk.sent_tokenize(text)
    #print(sentences)
    for sentence in sentences:
        words=nltk.word_tokenize(sentence)
        print(words)
        tagged=nltk.pos_tag(words)
        print(tagged)
```

```
being', 'NN'), ('and', 'CC'), ('happiness', 'NN'), ('of', 'IN'), ('thei
r', 'PRP$'), ('children', 'NNS'), ('.', '.')]
['A', 'mother', "'s", 'love', 'knows', 'no', 'bounds', ',', 'transcendi
ng', 'time', 'and', 'distance', '.']
[('A', 'DT'), ('mother', 'NN'), ("'s", 'POS'), ('love', 'NN'), ('know
s', 'VBZ'), ('no', 'DT'), ('bounds', 'NNS'), (',', ','), ('transcendin
g', 'VBG'), ('time', 'NN'), ('and', 'CC'), ('distance', 'NN'), ('.',
'.')]
['Her', 'wisdom', ',', 'resilience', ',', 'and', 'the', 'ability', 't
o', 'turn', 'ordinary', 'moments', 'into', 'cherished', 'memories', 'ma
ke', 'her', 'a', 'remarkable', 'figure', 'in', 'our', 'lives', '.']
[('Her', 'PRP$'), ('wisdom', 'NN'), (',', ','), ('resilience', 'NN'),
(',', ','), ('and', 'CC'), ('the', 'DT'), ('ability', 'NN'), ('to', 'T
O'), ('turn', 'VB'), ('ordinary', 'JJ'), ('moments', 'NNS'), ('into',
'IN'), ('cherished', 'JJ'), ('memories', 'NNS'), ('make', 'VBP'), ('he
r', 'PRP$'), ('a', 'DT'), ('remarkable', 'JJ'), ('figure', 'NN'), ('i
n', 'IN'), ('our', 'PRP$'), ('lives', 'NNS'), ('.', '.')]
['Whether', 'through', 'gentle', 'words', 'of', 'encouragement', 'a',
'warm', 'embrace', 'or', 'the', 'tireless', 'efforts', 'behind', 'the',
'scenes', 'a', 'mom', "'s", 'impact', 'is', 'immeasurable', 'shaping',
```

In [ ]: