Generic code for implementation of NLP steps

Importing Useful Libraries

In [1]:
```
import re
import nltk                                     # Used for performing the steps of NLP

from nltk import word_tokenize                  # For splitting strings into tokens
from nltk.probability import FreqDist           # Count the number of times that each token of an experiment occurs
from nltk.tokenize import blankline_tokenize    # Tokenize a string, treating any sequence of blank lines as a delimiter.

from nltk.stem import PorterStemmer             # Used for removing the "ing" endings from words
from nltk.stem import LancasterStemmer          # Used for removing the inflexional endings from words


from nltk.stem import WordNetLemmatizer         # Used for converting the word into its meaningful base form
from nltk.corpus import stopwords               # Used for removing the stop words

from nltk import ne_chunk                       # Used to perform grouping of tokens
```

For the string from the user

In [2]:
```
x= input("Enter the string: ")
```

```
Enter the string: It is very cold outside.
```

Step 01: Tokenization

In [3]:
```
x_token = word_tokenize(x)
f = FreqDist()
```

In [4]:
```
print(x_token)
```

```
['It', 'is', 'very', 'cold', 'outside', '.']
```

In [5]:
```
print("Number of tokens in the string: ", len(x_token))
```

```
Number of tokens in the string:  6
```

In [6]:
```
for word in x_token:
    f[word.lower()] = f[word.lower()] + 1
print(f)
print("The 10 most occuring tokens are:\n", f.most_common(10))
```

```
<FreqDist with 6 samples and 6 outcomes>
The 10 most occuring tokens are:
 [('it', 1), ('is', 1), ('very', 1), ('cold', 1), ('outside', 1), ('.', 1)]
```

In [7]:
```
x_blank = blankline_tokenize(x)
print("Number of blank lines within the string: ", len(x_blank))
```

```
Number of blank lines within the string:  1
```

In [8]:
```
x_bigrams = list(nltk.bigrams(x_token))
print(x_bigrams)
```

```
[('It', 'is'), ('is', 'very'), ('very', 'cold'), ('cold', 'outside'), ('outside', '.')]
```

In [9]:
```
x_trigrams = list(nltk.trigrams(x_token))
print(x_trigrams)
```

```
[('It', 'is', 'very'), ('is', 'very', 'cold'), ('very', 'cold', 'outside'), ('cold', 'outside', '.')]
```

In [10]:
```
x_ngrams = list(nltk.ngrams(x_token, 4))
print(x_ngrams)
```

```
[('It', 'is', 'very', 'cold'), ('is', 'very', 'cold', 'outside'), ('very', 'cold', 'outside', '.')]
```

Step 02: Stemming

In [11]:
```
ps = PorterStemmer()
for words in x_token:
    print(words + ": " + ps.stem(words))
```

```
It: It
is: is
very: veri
cold: cold
outside: outsid
.: .
```

In [12]:
```
ls = LancasterStemmer()
for words in x_token:
    print(words + ": " + ls.stem(words))
```

```
It: it
is: is
very: very
cold: cold
outside: outsid
.: .
```

Step 03: Lemmatization

In [13]:
```
lm = WordNetLemmatizer()
for words in x_token:
    print(words + ": " + lm.lemmatize(words))
```

```
It: It
is: is
very: very
cold: cold
outside: outside
.: .
```

Stopwords in English language

In [14]:
```
print(stopwords.words('english'))
print("\nNumber of stopwords in English language: ", len(stopwords.words('english')))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', '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', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', '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', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

Number of stopwords in English language:  179
```

Remove numbers and punctuation marks

In [15]:
```
punc = re.compile(r'[-.?!,:;()|0-9]')
post_punc = []
for words in x_token:
    word = punc.sub("", words)
    if len(word)>0:
        post_punc.append(word)

print(post_punc)
print("Length of string after removing numbers and punctuation marks: ", len(post_punc))
```

```
['It', 'is', 'very', 'cold', 'outside']
Length of string after removing numbers and punctuation marks:  5
```

Step 04: Part of Speech Tags & Named Entity Recognition

In [16]:
```
for token in x_token:
    print(nltk.pos_tag([token]))
```

```
[('It', 'PRP')]
[('is', 'VBZ')]
[('very', 'RB')]
[('cold', 'NN')]
[('outside', 'IN')]
[('.', '.')]
```

Step 05: Chunking

In [17]:
```
m = nltk.pos_tag(x_token)
n = ne_chunk(m)
print(n)
g = r"NP: {<DT>?<JJ>?<NN>}"
o = nltk.RegexpParser(g)
r = o.parse(m)
r.draw()
```

```
(S It/PRP is/VBZ very/RB cold/JJ outside/NN ./.)
```