Assignment 7A

In [1]:
```
import pandas as pd
import nltk
import re
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
```

In [2]:
```
text= "Tokenization is the first step in text analytics. The process of breaking down a text pa
```

In [3]:
```
#Sentence Tokenization
tokenized_text= sent_tokenize(text)
print(tokenized_text)
#Word Tokenization
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization is the first step in text analytics.', 'The process of breaking down a text para
graph into smaller chunks such as words or sentences is called Tokenization.']
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.', 'The', 'proces
s', 'of', 'breaking', 'down', 'a', 'text', 'paragraph', 'into', 'smaller', 'chunks', 'such', 'a
s', 'words', 'or', 'sentences', 'is', 'called', 'Tokenization', '.']
```

In [4]:
```
#Print stop words of English
stop_words=set(stopwords.words("english"))
print(stop_words)
text= "How to remove stop words with NLTK library in Python?"
text= re.sub('[^a-zA-Z]', ' ',text)
tokens = word_tokenize(text.lower())
filtered_text=[]
for w in tokens:
    if w not in stop_words:
        filtered_text.append(w)
print("Tokenized Sentence:",tokens)
print("Filterd Sentence:",filtered_text)
```

```
{"mightn't", 'both', 'theirs', 'at', 'it', 'hasn', "you'd", 'herself', 'o', 'nor', 'so', 'whil
e', 'didn', "wouldn't", 'where', 'her', "doesn't", 'very', 'mightn', 'between', 'in', 'have',
'if', 'now', 'of', 'd', "hadn't", 'haven', "shouldn't", 'a', 'own', 'y', "won't", 'to', 'do',
'them', 'ma', "that'll", 'no', 'by', 'this', 'ain', 'such', 'myself', 'further', 'but', "you'r
e", 'they', 'on', 'from', 'our', "didn't", 'he', 'couldn', 'these', "you'll", 'each', 'your',
't', 'my', 'will', 'she', 'above', 'more', 'aren', 'won', 're', "mustn't", 'which', 'below', 'a
n', 'some', 'whom', 'that', 'during', 'itself', "she's", "couldn't", 'me', 'be', 'does', 'all',
'until', 's', 'than', 'i', 'too', 'll', 'can', 'm', 'been', 'mustn', 'ours', "haven't", 'bein
g', 'few', 'was', 'under', 'its', 'out', 'weren', 'as', 'their', 'who', 'wouldn', 'doing', 'you
rself', "weren't", 'why', 'then', 'through', 'him', 'for', 'am', 'before', "wasn't", 'there',
'just', "you've", "aren't", 'up', "it's", 'should', 'and', "hasn't", 'again', 'because', 'othe
r', 'not', 'any', 'had', 'those', 'off', 'is', 'against', 'did', 'only', 'having', 'we', 'thems
elves', 'needn', 'shouldn', "shan't", 'yours', 'shan', 'after', 'once', "don't", 'when', 'are',
"needn't", 'most', 'hers', 'with', 'about', 'were', 'same', 'the', 'you', 'yourselves', 'has',
'his', "should've", 'down', 'here', 'or', 'wasn', 'how', 'himself', 'isn', 'hadn', 'don', 'does
n', 'over', 've', 'ourselves', "isn't", 'into', 'what'}
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'library', 'in',
'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'python']
```

In [5]:
```
#Stemming
e_words= ["wait", "waiting", "waited", "waits"]
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
print(rootWord)
```

```
wait
```

In [6]:
```
#Lemmatization
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
    print("Lemma for {} is {}".format(w,wordnet_lemmatizer.lemmatize(w)))
```

```
Lemma for studies is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
```

In [7]:
```
#Pos Tagging
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for word in words:
    print(nltk.pos_tag([word]))
```

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('her', 'PRP$')]
[('perfectly', 'RB')]
```

Assignment 7B

In [8]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import math
```

In [9]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
    numOfWordsA[word] += 1
    numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
    numOfWordsB[word] += 1
```

In [10]:
```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
```

In [11]:
```
def computeIDF(documents):

    N = len(documents)
    idfDict = dict.fromkeys(documents[0].keys(), 0)
    for document in documents:
        for word, val in document.items():
            if val > 0:
                idfDict[word] += 1
    for word, val in idfDict.items():
        idfDict[word] = math.log(N / float(val))
    return idfDict
idfs = computeIDF([numOfWordsA, numOfWordsB])
idfs
```

Out[11]:

```
{'the': 0.0,
 'fourth': 0.6931471805599453,
 'is': 0.0,
 'Mars': 0.6931471805599453,
 'Sun': 0.6931471805599453,
 'Planet': 0.6931471805599453,
 'Jupiter': 0.6931471805599453,
 'from': 0.6931471805599453,
 'planet': 0.6931471805599453,
 'largest': 0.6931471805599453}
```

In [12]:

```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf
tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
df = pd.DataFrame([tfidfA, tfidfB])
df
```

Out[12]:

	the	fourth	is	Mars	Sun	Planet	Jupiter	from	planet	largest
0	0.0	0.000000	0.0	0.000000	0.000000	0.138629	0.138629	0.000000	0.000000	0.138629
1	0.0	0.086643	0.0	0.086643	0.086643	0.000000	0.000000	0.086643	0.086643	0.000000

In [ ]: