Assignment No. 7

Aim : Text Analytics

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

Code :

In [4]:
```
import nltk
```

In [5]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Welcome\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[5]: True

In [6]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Welcome\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[6]: True

In [7]:
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Welcome\AppData\Roaming\nltk_data...
```

Out[7]: True

In [9]:
```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Welcome\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[9]: True

In [10]:
```
text= "Tokenization is the first step in text analytics. The process of breaking down a text pa
text
```

Out[10]:
```
'Tokenization is the first step in text analytics. The process of breaking down a text paragraph i
nto smaller chunkssuch as words or sentences is called Tokenization.'
```

In [11]:
```
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)
```

```
['Tokenization is the first step in text analytics.', 'The process of breaking down a text paragra
ph into smaller chunkssuch as words or sentences is called Tokenization.']
```

In [12]:
```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.', 'The', 'process',
'of', 'breaking', 'down', 'a', 'text', 'paragraph', 'into', 'smaller', 'chunkssuch', 'as', 'word
s', 'or', 'sentences', 'is', 'called', 'Tokenization', '.']
```

In [17]:
```
import re
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
{'aren', "you're", 'isn', 'didn', 'and', 'those', "we'll", "they're", "weren't", 'about', 'needn',
'as', "they'd", 'below', "should've", 'am', 'we', 've', 'very', 'each', 'ma', 'under', 'a', "are
n't", 'yourselves', 'herself', 'these', 'wasn', "won't", "didn't", 'why', "you'd", "don't", "i'v
e", "i'll", "i'm", 'mustn', 'own', "they'll", 'won', 'him', 'shan', 'she', "that'll", 'them', 'whi
ch', 'your', 'for', 'it', "it'd", "mustn't", 'but', 'ain', 'because', 'now', 'at', "they've", 'som
e', "she'd", "she'll", 'after', 'ours', "we've", 'only', "you'll", 'of', 'myself', 'who', 'weren',
'my', 'doing', 'haven', 'he', 'me', 'over', "shouldn't", 'just', "i'd", "it's", 'd', 'did', 'don',
'before', 'if', 'have', 're', 'couldn', 'our', 'is', 'there', "it'll", "hadn't", 'his', 'by', "doe
sn't", 'how', 'same', "we'd", "he'll", 'o', 'no', 'do', 'most', 'was', "you've", 'so', "isn't", "h
asn't", 'can', 'more', 'being', "needn't", 'out', 'than', 'were', 'during', 'its', 'above', 'are',
'you', 'both', 'with', 'not', 'been', 'further', 'here', 'in', 'their', 'to', 'where', 'down', "mi
ghtn't", 'has', 'mightn', 'll', 'having', 'theirs', "we're", "couldn't", 'into', 'an', 'yourself',
's', 'themselves', 'when', 'wouldn', "she's", 'itself', 'up', 'while', 'be', 'from', 'whom', 'wil
l', 'shouldn', 'had', 'against', "he'd", "wouldn't", 'such', 'again', 'the', 'through', 'off',
'y', 'this', 'other', 'hadn', 'hers', 'they', 'yours', 'on', "haven't", 'i', 'doesn', 'few', 'he
r', 'himself', 'all', 'then', 'that', "wasn't", 'or', "he's", 'should', 't', 'between', 'm', 'unti
l', 'ourselves', 'nor', 'what', 'once', "shan't", 'too', 'hasn', 'any', 'does'}
```

In [18]:
```
text= "How to remove stop words with NLTK library inPython?"
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)
```

```
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'library', 'inpytho
n']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'inpython']
```

In [20]:
```
from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited","waits"]
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
print(rootWord)
```

```
wait
```

In [25]:
```
from nltk.stem import WordNetLemmatizer
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 [29]:
```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit herperfectly"
words=word_tokenize(data)
print(data)
print(words)
```

```
The pink sweater fit herperfectly
['The', 'pink', 'sweater', 'fit', 'herperfectly']
```

In [30]:
```
for word in words:
    print(nltk.pos_tag([word]))

```

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('herperfectly', 'RB')]
```

In [31]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [32]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
print(documentA)
print(documentB)
```

```
Jupiter is the largest Planet
Mars is the fourth planet from the Sun
```

In [34]:
```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
print(bagOfWordsA)
print(bagOfWordsB)
```

```
['Jupiter', 'is', 'the', 'largest', 'Planet']
['Mars', 'is', 'the', 'fourth', 'planet', 'from', 'the', 'Sun']
```

In [36]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

In [37]:
```
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 [47]:
```
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 [48]:
```
def computeIDF(documents):
    import math
    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
```

In [49]:
```
idfs = computeIDF([numOfWordsA, numOfWordsB])
idfs
```

Out[49]:
```
{'Jupiter': 0.6931471805599453,
 'planet': 0.6931471805599453,
 'largest': 0.6931471805599453,
 'is': 0.0,
 'from': 0.6931471805599453,
 'fourth': 0.6931471805599453,
 'Planet': 0.6931471805599453,
 'Mars': 0.6931471805599453,
 'Sun': 0.6931471805599453,
 'the': 0.0}
```

In [50]:
```
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[50]:

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

Name : Mohan Kadambande

Roll No. : 13212 (TECO-b1)

In [ ]:
```

```