```
Requirement already satisfied: nltk in c:\programdata\anaconda3\lib\site-p
```

In [1]:

```
pip install nltk
```

```
ackages (3.8.1)
Requirement already satisfied: click in
c:\programdata\anaconda3\lib\sitepackages (from nltk) (8.0.4)
Requirement already satisfied: joblib in c:\programdata\anaconda3\lib\site
-packages (from nltk) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\programdata\anaconda3
\lib\site-packages (from nltk) (2022.7.9)
Requirement already satisfied: tqdm in c:\programdata\anaconda3\lib\site-p
ackages (from nltk) (4.65.0)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\si
te-packages (from click->nltk) (0.4.6) Note: you may need to restart the
kernel to use updated packages.
```

In [1]:

```
import nltk as nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to [nltk_data]
C:\Users\Ayush\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package stopwords to [nltk_data]
C:\Users\Ayush\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package wordnet to [nltk_data]
C:\Users\Ayush\AppData\Roaming\nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Ayush\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[1]: True

In [3]:

```
text= "Tokenization is the first step in text analytics. The process of brea
```

In [4]:

```
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 brea
king down a text paragraph into smaller chunkssuch as words or sentences i
s called Tokenization.']
```

In [5]:

```
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', 'words', 'or', 'sentences', 'is',
'called', 'Tokenization', '.']
```

In [7]:
```
import regex as re
from nltk.corpus import stopwords
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)
```

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

In [8]:
```
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 [9]:
```
import nltk
from nltk.tokenize import word_tokenize
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')]
```

In [10]:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [11]:

```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

In [12]:

```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

In [13]:

```
uniqueWords =set(bagOfWordsA).union(set(bagOfWordsB))
```

In [15]:

```
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 [19]:

```
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 [20]:

```
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
idfs = computeIDF([numOfWordsA,numOfWordsB])
idfs
```

Out[20]:

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

In [21]:

```
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[21]:

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

# Tanmay Dixit TE13143