In [1]:

```
pip install nltk
```

```
Defaulting to user installation because normal site-packages is not wr
iteable
Requirement already satisfied: nltk in /home/ubuntu/.local/lib/python
3.8/site-packages (3.8.1)
Requirement already satisfied: click in /usr/lib/python3/dist-packages
(from nltk) (7.0)
Requirement already satisfied: tqdm in /home/ubuntu/.local/lib/python
3.8/site-packages (from nltk) (4.64.1)
Requirement already satisfied: regex>=2021.8.3 in /home/ubuntu/.local/
lib/python3.8/site-packages (from nltk) (2023.3.23)
Requirement already satisfied: joblib in /home/ubuntu/.local/lib/pytho
n3.8/site-packages (from nltk) (1.2.0)
WARNING: You are using pip version 22.0.4; however, version 23.1.2 is
 available.
You should consider upgrading via the '/usr/bin/python3 -m pip install
--upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

In [2]:

```
import nltk
```

In [3]:

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

```
[nltk_data] Downloading package punkt to /home/ubuntu/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/ubuntu/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/ubuntu/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/ubuntu/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[3]:

```
True
```

In [4]:

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

In [5]:

```
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 paragraph into smaller chunks Ssuch as words or s
entences is called Tokenization.']
```

In [6]:

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytic
s', '.', 'The', 'process', 'of', 'breaking', 'down', 'a', 'text', 'par
agraph', 'into', 'smaller', 'chunks', 'Ssuch', 'as', 'words', 'or', 's
entences', 'is', 'called', 'Tokenization', '.']
```

In [7]:

```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

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

In [8]:

```
import re
```

In [9]:

```
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)
```

```
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with',
'nltk', 'library', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'pyth
on']
```

In [10]:

```
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 [11]:

```
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 [12]:

```
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 [13]:

```
import nltk
paragraph =  """I have three visions for India. In 3000 years of our history, peopl
               the world have come and invaded us, captured our lands, conquered ou
               Yet we have not done this to any other nation. We have not conquered
               We have not grabbed their land, their culture,
               their history and tried to enforce our way of life on them.
               Why? """
```

In [14]:

```
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

In [15]:

```
wn=WordNetLemmatizer()
sentences=nltk.sent_tokenize(paragraph)
```

In [16]:

```
corpus=[]
for i in range(len(sentences)):
    review=re.sub('[^a-zA-Z]',' ',sentences[i])
    review=review.lower()
    review=review.split()
    review=[wn.lemmatize(word) for word in review if not word in set(stopwords.word
    review=' '.join(review)
    corpus.append(review)
corpus
```

Out[16]:

```
['three vision india',
 'year history people world come invaded u captured land conquered min
d',
 'yet done nation',
 'conquered anyone',
 'grabbed land culture history tried enforce way life',
 '']
```

In [17]:

```
# Creating the TF-IDF model
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer()
X=tf.fit_transform(corpus).toarray()
print(X)
```

```
[[0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.57735027 0.         0.
  0.         0.         0.         0.         0.57735027 0.
  0.57735027 0.         0.         0.         0.        ]
 [0.         0.33301397 0.33301397 0.27307622 0.         0.
  0.         0.         0.27307622 0.         0.33301397 0.27307622
  0.         0.33301397 0.         0.33301397 0.         0.
  0.         0.         0.33301397 0.33301397 0.        ]
 [0.         0.         0.         0.         0.         0.57735027
  0.         0.         0.         0.         0.         0.
  0.         0.         0.57735027 0.         0.         0.
  0.         0.         0.         0.         0.57735027]
 [0.77326237 0.         0.         0.6340862  0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.36898493 0.
  0.36898493 0.36898493 0.30257292 0.         0.         0.30257292
  0.36898493 0.         0.         0.         0.         0.36898493
  0.         0.36898493 0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.        ]]
```

In [ ]:

```

```