```
In [1]: pip install nltk
```

```
Requirement already satisfied: nltk in d:\anaconda\lib\site-packages (3.9.1)
Requirement already satisfied: click in d:\anaconda\lib\site-packages (from
nltk) (8.1.7)
Requirement already satisfied: joblib in d:\anaconda\lib\site-packages (from
nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in d:\anaconda\lib\site-packa
ges (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in d:\anaconda\lib\site-packages (from n
ltk) (4.66.5)
Requirement already satisfied: colorama in d:\anaconda\lib\site-packages (fr
om click->nltk) (0.4.6)
Note: you may need to restart the kernel to use updated packages.
```

```
In [2]: import nltk
        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
        import re
        from sklearn.feature_extraction.text import TfidfVectorizer
```

```
In [3]: nltk.download('punkt')
        nltk.download('stopwords')
        nltk.download('wordnet')
        nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to C:\Users\VEDIKA/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\VEDIKA/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\VEDIKA/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\VEDIKA/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

```
Out[3]: True
```

```
In [4]: text="Tokenization replaces a sensitive data element, for example, a bank ac
```

```
In [5]: sentences = sent_tokenize(text)
        print(sentences)
```

```
['Tokenization replaces a sensitive data element, for example, a bank accoun
t number, with a non-sensitive substitute, known as a token.', 'The token is
a randomized data string that has no essential or exploitable value or meani
ng.', 'It is a unique identifier which retains all the pertinent information
about the data without compromising its security']
```

In [6]:
```
words = word_tokenize(text)
print(words)
```

```
['Tokenization', 'replaces', 'a', 'sensitive', 'data', 'element', ',', 'fo
r', 'example', ',', 'a', 'bank', 'account', 'number', ',', 'with', 'a', 'non
-sensitive', 'substitute', ',', 'known', 'as', 'a', 'token', '.', 'The', 'to
ken', 'is', 'a', 'randomized', 'data', 'string', 'that', 'has', 'no', 'essen
tial', 'or', 'exploitable', 'value', 'or', 'meaning', '.', 'It', 'is', 'a',
'unique', 'identifier', 'which', 'retains', 'all', 'the', 'pertinent', 'info
rmation', 'about', 'the', 'data', 'without', 'compromising', 'its', 'securit
y']
```

In [7]:
```
stop_words = set(stopwords.words('english'))
print(stop_words)
```

```
{'should', "didn't", 'weren', 'be', 'hers', 'they', 'no', 'shan', 'won', 'yo
urs', 'yourselves', 'after', 's', 'to', "wasn't", "weren't", 'its', 'themsel
ves', 'same', 'o', 'being', 'how', 't', 'ain', 'aren', 'have', 'about', 'wa
s', 'do', 'until', 'did', 'doesn', 'down', 'it', 'most', 'himself', 'very',
'the', 'of', 'our', 'i', 'such', "hadn't", "you're", 'if', 'ma', 'off', 'abo
ve', 'who', 'not', 'further', "won't", 'itself', 'few', 'is', "you'd", 'doe
s', 'as', 'with', 'once', 'isn', 'herself', 'those', 'all', 'were', 'don',
'below', 'both', 'too', 'why', 'some', 'during', 'are', 'whom', "aren't", 'j
ust', "you've", 'yourself', 'am', "doesn't", "it's", 'when', 'while', 'unde
r', 'been', 'myself', 'then', "don't", "wouldn't", 'she', 'we', 'more', 'u
p', 'your', 'and', 'against', 'out', 'each', "hasn't", 'any', 'own', 'what',
'you', 'again', 'at', 'him', 'only', "that'll", 'from', 'couldn', 'than', 'h
aving', 'ours', 'my', 'that', 'before', 'he', 'but', "needn't", 'or', 'did
n', 'ourselves', "you'll", 'haven', 'between', 'wasn', "she's", "mightn't",
'their', 'so', 'can', 'a', 'through', 'now', 'here', "isn't", 'wouldn', 'm
e', 'into', 'where', 'for', 'will', 'm', 'needn', "shouldn't", 'y', 'an', 'm
ightn', 'other', 'had', "should've", 'doing', 'because', "mustn't", 'their
s', 're', 'in', 'by', 'on', 'hadn', "shan't", 'll', "couldn't", 'hasn', 'sho
uldn', 'has', 've', 'nor', 'her', 'these', 'them', 'which', 'mustn', 'd', "h
aven't", 'over', 'there', 'his', 'this'}
```

In [8]:
```
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("Filtered Sentence:", filtered_text)
```

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

In [9]:
```
ps = PorterStemmer()
stemmed_words = ['wait', 'waiting', 'waited', 'waits']
for w in stemmed_words:
    print(w, ":", ps.stem(w))
print("Original Sentence:", stemmed_words)
```

```
wait : wait
waiting : wait
waited : wait
waits : wait
Original Sentence: ['wait', 'waiting', 'waited', 'waits']
```

In [10]:
```
lemmatizer = WordNetLemmatizer()
text= "studies studying cies cry"
tokenization= nltk.word_tokenize(text)
for w in tokenization:
    print("Lemma for {} is {}".format(w, lemmatizer.lemmatize(w)))
```

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

In [11]:
```
data = 'The pink sweater fits her perfectly'
words = nltk.word_tokenize(data)
for word in words:
    print(nltk.pos_tag([word]))
```

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

In [12]:
```
paragraph= """India is a vast country with second highest populati
on in the world. It is a country
with diverse cultures, traditions and beliefs. People in India cel
ebrate unity in diversity.
Festivals like Diwali, Holi, Navratri, Ramzan, Christmas etc. are
celebrated by people across India
and create a sense of brotherhood and cultural unity. Each festiva
l has its religious and cultural importance."""
```

In [13]:
```
wn = WordNetLemmatizer()
sentences = nltk.sent_tokenize(paragraph)
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(stopw
    review = ' '.join(review)
    corpus.append(review)
print(corpus)
```

```
['india vast country second highest populati world', 'country diverse cultur
e tradition belief', 'people india cel ebrate unity diversity', 'festival li
ke diwali holi navratri ramzan christmas etc', 'celebrated people across ind
ia create sense brotherhood cultural unity', 'festiva l religious cultural i
mportance']
```

```
In [14]: tfidf = TfidfVectorizer()
         X = tfidf.fit_transform(corpus).toarray()
         print(X)
```

```
[[0.         0.         0.         0.         0.         0.
  0.33061545 0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.40318254
  0.         0.         0.27912828 0.         0.         0.
  0.40318254 0.         0.         0.40318254 0.         0.
  0.         0.40318254 0.40318254]
 [0.         0.46262479 0.         0.         0.         0.
  0.37935895 0.         0.         0.46262479 0.46262479 0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.46262479
  0.         0.         0.        ]
 [0.         0.         0.         0.45529187 0.         0.
  0.         0.         0.         0.         0.         0.45529187
  0.         0.45529187 0.         0.         0.         0.
  0.         0.         0.31520422 0.         0.         0.37334585
  0.         0.         0.         0.         0.         0.
  0.37334585 0.         0.        ]
 [0.         0.         0.         0.         0.         0.35355339
  0.         0.         0.         0.         0.         0.
  0.35355339 0.         0.35355339 0.         0.35355339 0.
  0.35355339 0.         0.         0.35355339 0.35355339 0.
  0.         0.35355339 0.         0.         0.         0.
  0.         0.         0.        ]
 [0.36523197 0.         0.36523197 0.         0.36523197 0.
  0.         0.36523197 0.29949544 0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.25285463 0.         0.         0.29949544
  0.         0.         0.         0.         0.36523197 0.
  0.29949544 0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.42790272 0.         0.         0.
  0.         0.         0.         0.52182349 0.         0.
  0.         0.52182349 0.         0.         0.         0.
  0.         0.         0.52182349 0.         0.         0.
  0.         0.         0.        ]]
```