In [1]:

```python
import 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\Welcome\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Welcome\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Welcome\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[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[1]:

```
True
```

```
#PERFORMING STEMMING
```

In [2]:

```python
from nltk.stem import PorterStemmer
ps = PorterStemmer()
```

In [3]:

```python
e_words = ["wait", "waiting", "waited", "waits"]
```

In [9]:

```python
for i in e_words:
    rootWord = ps.stem(i)
    print(f"{i} -> {rootWord}")
```

```
wait -> wait
waiting -> wait
waited -> wait
waits -> wait
```

In [ ]:

```python
#PERFORMING  LEMMATIZATION
```

In [11]:

```python
from nltk.stem import WordNetLemmatizer
wl = WordNetLemmatizer()
from nltk.tokenize import word_tokenize
```

In [12]:

```python
text = "studies studying cries cry"
```

In [14]:

```python
tokenized_words = word_tokenize(text)
print("Tokenized words:", tokenized_words)
```

```
Tokenized words: ['studies', 'studying', 'cries', 'cry']
```

In [15]:

```python
for i in tokenized_words:
    lemma = wl.lemmatize(i)
    print(f"Lemma for {i}: {lemma}")
```

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

```python
#POS TAGGING
```

In [17]:

```python
data = "The pink sweater fit her perfectly"
words = word_tokenize(data)
```

In [18]:

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

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

```python
#Algorithm for Create representation of document by calculating TFIDF
```

In [19]:

```python
import pandas as pd
import math
```

In [21]:

```python
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
xA = documentA.split(' ')
xB = documentB.split(' ')
```

In [24]:

```python
uniqueWords = set(xA).union(set(xB))
print(uniqueWords)
```

```
{'is', 'largest', 'from', 'Jupiter', 'fourth', 'Sun', 'planet', 'the', 'Ma
rs', 'Planet'}
```

In [25]:

```python
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in xA:
    numOfWordsA[word] += 1
```

In [26]:

```python
numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in xB:
    numOfWordsB[word] += 1
```

In [27]:

```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict

tfA = computeTF(numOfWordsA, xA)
tfB = computeTF(numOfWordsB, xB)
```

In [28]:

```
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)) if val > 0 else 0
    return idfDict

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

In [29]:

```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf

```

In [30]:

```
tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
```

In [31]:

```
df = pd.DataFrame([tfidfA, tfidfB], index=['Document A', 'Document B'])
print(df)
```

```
             is   largest      from   Jupiter    fourth       Sun     plane
t  \
Document A  0.0  0.138629  0.000000  0.138629  0.000000  0.000000  0.00000
0
Document B  0.0  0.000000  0.086643  0.000000  0.086643  0.086643  0.08664
3

            the      Mars    Planet
Document A  0.0  0.000000  0.138629
Document B  0.0  0.086643  0.000000
```

In [ ]:

```
             NAME: NEHA JADHAV
             ROLL NO: 13247
```

In [ ]:

```

```

In [ ]:

```
```