In [1]:
```
!pip install nltk
!pip install pandas
!pip install scikit-learn
```

```
Requirement already satisfied: nltk in /home/sargam/.conda/envs/myenv/lib/
python3.11/site-packages (3.9.1)
Requirement already satisfied: click in /home/sargam/.conda/envs/myenv/li
b/python3.11/site-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /home/sargam/.conda/envs/myenv/li
b/python3.11/site-packages (from nltk) (1.5.0)
Requirement already satisfied: regex>=2021.8.3 in /home/sargam/.conda/env
s/myenv/lib/python3.11/site-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /home/sargam/.conda/envs/myenv/lib/
python3.11/site-packages (from nltk) (4.67.1)
Requirement already satisfied: pandas in /home/sargam/.conda/envs/myenv/li
b/python3.11/site-packages (2.2.3)
Requirement already satisfied: numpy>=1.23.2 in /home/sargam/.conda/envs/m
yenv/lib/python3.11/site-packages (from pandas) (2.0.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/sargam/.con
da/envs/myenv/lib/python3.11/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /home/sargam/.conda/envs/my
env/lib/python3.11/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /home/sargam/.conda/envs/
myenv/lib/python3.11/site-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /home/sargam/.conda/envs/myenv/
lib/python3.11/site-packages (from python-dateutil>=2.8.2->pandas) (1.17.
0)
Requirement already satisfied: scikit-learn in /home/sargam/.conda/envs/my
env/lib/python3.11/site-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /home/sargam/.conda/envs/m
yenv/lib/python3.11/site-packages (from scikit-learn) (2.0.1)
Requirement already satisfied: scipy>=1.6.0 in /home/sargam/.conda/envs/my
env/lib/python3.11/site-packages (from scikit-learn) (1.15.2)
Requirement already satisfied: joblib>=1.2.0 in /home/sargam/.conda/envs/m
yenv/lib/python3.11/site-packages (from scikit-learn) (1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in /home/sargam/.cond
a/envs/myenv/lib/python3.11/site-packages (from scikit-learn) (3.6.0)
```

In [4]:
```
import nltk
import re
import math
import pandas as pd
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [18]:
```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
```

```
[nltk_data] Downloading package punkt to /home/sargam/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/sargam/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/sargam/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/sargam/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package punkt_tab to /home/sargam/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /home/sargam/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
```

Out[18]: True

In [6]:
```
text = "Tokenization is the first step in text analytics. The process of
```

In [9]:
```
tokenized_text = sent_tokenize(text)
print("Sentence Tokenization:", tokenized_text)
```

```
Sentence Tokenization: ['Tokenization is the first step in text analytic
s.', 'The process of breaking down a text paragraph into smaller chunks su
ch as words or sentences is called Tokenization.']
```

In [10]:
```
tokenized_word = word_tokenize(text)
print("Word Tokenization:", tokenized_word)
```

```
Word Tokenization: ['Tokenization', 'is', 'the', 'first', 'step', 'in', 't
ext', 'analytics', '.', 'The', 'process', 'of', 'breaking', 'down', 'a',
'text', 'paragraph', 'into', 'smaller', 'chunks', 'such', 'as', 'words',
'or', 'sentences', 'is', 'called', 'Tokenization', '.']
```

In [11]:
```
stop_words = set(stopwords.words("english"))
print("Stop Words:", stop_words)
```

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

In [12]:
```
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 = [w for w in tokens if w not in stop_words]
```

In [13]:
```
print("Tokenized Sentence:", tokens)
print("Filtered Sentence:", filtered_text)
```

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

In [14]:
```
e_words = ["wait", "waiting", "waited", "waits"]
ps = PorterStemmer()
print("\nStemming Results:")
for w in e_words:
    root_word = ps.stem(w)
    print(f"Original: {w} -> Stemmed: {root_word}")
```

```
Stemming Results:
Original: wait -> Stemmed: wait
Original: waiting -> Stemmed: wait
Original: waited -> Stemmed: wait
Original: waits -> Stemmed: wait
```

In [15]:
```
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
```

In [16]:
```
print("\nLemmatization Results:")
for w in tokenization:
    print(f"Lemma for {w} is {wordnet_lemmatizer.lemmatize(w)}")
```

```
Lemmatization Results:
Lemma for studies is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
```

In [19]:
```
data = "The pink sweater fit her perfectly"
words = word_tokenize(data)
print("\nPOS Tagging:")
for word in words:
    print(f"{word}: {nltk.pos_tag([word])}")
```

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

In [20]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

In [21]:
```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

In [22]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

In [23]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
    numOfWordsA[word] += 1
```

In [24]:
```
numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
    numOfWordsB[word] += 1
```

In [25]:
```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict
```

In [26]:
```
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
```

In [27]:
```
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))
    return idfDict

idfs = computeIDF([numOfWordsA, numOfWordsB])
print("\nInverse Document Frequency (IDF) Values:", idfs)
```

```
Inverse Document Frequency (IDF) Values: {'Planet': 0.6931471805599453, 'f
rom': 0.6931471805599453, 'the': 0.0, 'fourth': 0.6931471805599453, 'large
st': 0.6931471805599453, 'Sun': 0.6931471805599453, 'is': 0.0, 'Jupiter':
0.6931471805599453, 'planet': 0.6931471805599453, 'Mars': 0.69314718055994
53}
```

In [28]:
```
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])
print("\nTF-IDF Values for Documents A and B:")
print(df)
```

```
TF-IDF Values for Documents A and B:
     Planet      from  the    fourth   largest       Sun   is   Jupiter  \
0  0.138629  0.000000  0.0  0.000000  0.138629  0.000000  0.0  0.138629
1  0.000000  0.086643  0.0  0.086643  0.000000  0.086643  0.0  0.000000

     planet      Mars
0  0.000000  0.000000
1  0.086643  0.086643
```

In [ ]: