In [ ]:
```
                                ASSIGNMENT 07
```

In [ ]:
```
1. Basic concepts of Text Analytics
2. Text Analysis Operations using natural language toolkit
3. Text Analysis Model using TF-IDF.
4. Bag of Words (BoW)
```

In [1]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /home/ca0567cd-8494-4349-bf55-
[nltk_data]     d48912f18f72/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/ca0567cd-8494-4349-bf55-
[nltk_data]     d48912f18f72/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /home/ca0567cd-8494-4349-bf55-
[nltk_data]     d48912f18f72/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/ca0567cd-8494-4349-bf55-
[nltk_data]     d48912f18f72/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[1]: True

In [2]:
```
text = "Tokenization is the first step in text analytics. The process of breaking d
```

In [3]:
```
from nltk.tokenize import sent_tokenize, word_tokenize

tokenized_text = sent_tokenize(text)
print("Sentence Tokenization:")
print(tokenized_text)


tokenized_word = word_tokenize(text)
print("\nWord Tokenization:")
print(tokenized_word)
```

```
Sentence Tokenization:
['Tokenization is the first step in text analytics.', 'The process of breaking down
a text paragraph into smaller chunks such as words or sentences is called Tokenizati
on.']

Word Tokenization:
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.', 'Th
e', 'process', 'of', 'breaking', 'down', 'a', 'text', 'paragraph', 'into', 'smalle
r', 'chunks', 'such', 'as', 'words', 'or', 'sentences', 'is', 'called', 'Tokenizatio
n', '.']
```

In [4]:
```python
import re
from nltk.corpus import stopwords
import string

stop_words = set(stopwords.words("english"))
print("\nStopwords in 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 = [w for w in tokens if w not in stop_words]
print("\nTokenized Sentence:", tokens)
print("Filtered Sentence:", filtered_text)
```

```
Stopwords in English:
{'other', 'than', 'against', 'how', "he's", 'here', 'has', 'so', 'are', 'your', 'wer
en', 'when', 'an', 'not', "needn't", 'didn', 'am', 'can', 'myself', 'shouldn', "shou
ldn't", 'ma', 'he', "aren't", 'as', "she'll", 'what', "they've", 'of', 'over', 'bee
n', 'after', 'same', "they'd", 'those', 'why', "wasn't", 'the', "didn't", "might
n't", 'its', 'a', 'these', "we'd", 'into', "it'll", "doesn't", 'doing', 'won', 'an
d', "that'll", "he'll", 'y', 'few', 'where', 'some', 'between', 'she', 'which', 'und
er', 'does', 'o', 'once', 'very', 'doesn', 'should', 'couldn', 'during', 'to', 'at',
'each', "we've", "i've", 'll', "we're", 'their', 'i', 'was', 'were', 'needn', 've',
'isn', 'only', 'with', 'don', "couldn't", "hadn't", 'further', 'did', 'her', "must
n't", 'had', 'no', 'they', 'until', 'itself', "he'd", 'hasn', 'wasn', 'whom', 'herse
lf', 'from', 'ours', 'all', 'while', 'will', "you'll", 'own', "don't", "wouldn't",
'in', "they're", 'before', 'such', 'hers', 'nor', 'that', 'both', 'any', 'themselve
s', 'being', 'theirs', "hasn't", 'having', 'be', 'ourselves', "you're", "i'm", "shou
ld've", "i'll", 'up', "she's", 'now', "it's", 'on', "shan't", 'haven', "weren't", "w
on't", 'down', 'have', 'him', 'it', "haven't", "we'll", 'our', 'again', "she'd", 'th
is', 'too', 'aren', 'hadn', 'ain', 'yourself', 'through', 'most', 'off', 'because',
'then', 'who', 'yourselves', 'do', 'out', 'wouldn', 'about', 'but', 'mustn', 'ther
e', 'you', "i'd", 't', 'below', 'by', 'or', 'we', 're', 'is', 'm', "they'll", "you'v
e", 'just', "it'd", 'yours', 'them', 'd', 'himself', 'his', 's', 'if', 'me', 'more',
'above', 'shan', "you'd", "isn't", 'my', 'mightn', 'for'}

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

In [5]:
```python
from nltk.stem import PorterStemmer
```

```
e_words = ["wait", "waiting", "waited", "waits"]
ps = PorterStemmer()

print("\nStemming:")
for w in e_words:
    rootWord = ps.stem(w)
    print(f"Original: {w}, Stemmed: {rootWord}")
```

```
Stemming:
Original: wait, Stemmed: wait
Original: waiting, Stemmed: wait
Original: waited, Stemmed: wait
Original: waits, Stemmed: wait
```

In [6]:
```
from nltk.stem import WordNetLemmatizer


wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokens = nltk.word_tokenize(text)

print("\nLemmatization:")
for w in tokens:
    lemma = wordnet_lemmatizer.lemmatize(w)
    print(f"Lemma for {w}: {lemma}")
```

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

In [7]:
```
from nltk.tokenize import word_tokenize

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

print("\nPOS Tagging:")
for word in words:
    print(nltk.pos_tag([word]))
```

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

In [9]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

In [10]:
```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

In [11]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

In [12]:
```
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 [13]:
```
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 [14]:
```
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)) if val > 0 else 0
    return idfDict

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

In [15]:
```
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)
```

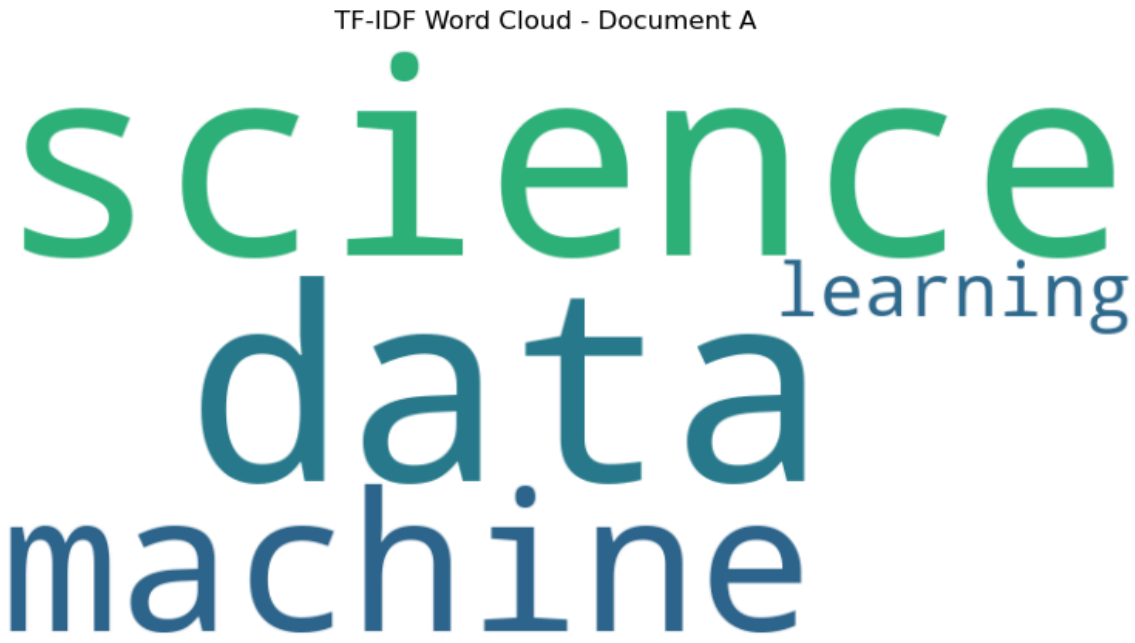
In [16]:
```
df = pd.DataFrame([tfidfA, tfidfB], index=['DocumentA', 'DocumentB'])
print(df)
```

```
            Jupiter  the   is       Sun      from    planet    Planet  \
DocumentA  0.138629  0.0  0.0  0.000000  0.000000  0.000000  0.138629
DocumentB  0.000000  0.0  0.0  0.086643  0.086643  0.086643  0.000000

               Mars   largest    fourth
DocumentA  0.000000  0.138629  0.000000
DocumentB  0.086643  0.000000  0.086643
```

In [8]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
 # Sample TF and IDF values (replace with your actual data)
tfA = {'data': 0.2, 'science': 0.4, 'machine': 0.1, 'learning': 0.3}
idfs = {'data': 1.5, 'science': 1.2, 'machine': 1.8, 'learning': 1.3}
 # Compute TF-IDF
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs.get(word, 0)
    return tfidf
tfidfA = computeTFIDF(tfA, idfs)
 # Generate Word Cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(' '

 # Plot it
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('TF-IDF Word Cloud - Document A')
plt.show()
```



In [ ]:
```
NAME: Karan More
ROLLNO:13234
```