# Assignment No. 7

Aim : Text Analytics

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

Code :

In [1]:
```
import nltk
```

In [2]:
```
nltk.download('punkt')
```

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

Out[2]: True

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

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

Out[3]: True

In [4]:
```
nltk.download('wordnet')
```

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

Out[4]: True

In [5]:
```
nltk.download('averaged_perceptron_tagger')
```

```
[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[5]: True

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

Out[6]: 'Tokenization is the first step in text analytics. The process of breaking down a text paragraph i
nto smaller chunkssuch as words or sentences is called Tokenization.'

In [7]:
```
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 paragra
ph into smaller chunkssuch as words or sentences is called Tokenization.']
```

In [8]:
```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.', 'The', 'process',
'of', 'breaking', 'down', 'a', 'text', 'paragraph', 'into', 'smaller', 'chunkssuch', 'as', 'word
s', 'or', 'sentences', 'is', 'called', 'Tokenization', '.']
```

In [9]:
```
import re
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

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

In [10]:
```
text= "How to remove stop words with NLTK library inPython?"
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', 'inpytho
n']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'inpython']
```

In [11]:
```
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 [12]:
```
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 [13]:
```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit herperfectly"
words=word_tokenize(data)
print(data)
print(words)
```

```
The pink sweater fit herperfectly
['The', 'pink', 'sweater', 'fit', 'herperfectly']
```

In [14]:
```
for word in words:
    print(nltk.pos_tag([word]))

```

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('herperfectly', 'RB')]
```

In [15]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [16]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
print(documentA)
print(documentB)
```

```
Jupiter is the largest Planet
Mars is the fourth planet from the Sun
```

In [17]:
```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
print(bagOfWordsA)
print(bagOfWordsB)
```

```
['Jupiter', 'is', 'the', 'largest', 'Planet']
['Mars', 'is', 'the', 'fourth', 'planet', 'from', 'the', 'Sun']
```

In [18]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

In [19]:
```
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 [20]:
```
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 [21]:
```
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))
    return idfDict
```

In [22]:
```
idfs = computeIDF([numOfWordsA, numOfWordsB])
idfs
```

Out[22]:
```
{'fourth': 0.6931471805599453,
 'Sun': 0.6931471805599453,
 'Planet': 0.6931471805599453,
 'largest': 0.6931471805599453,
 'Mars': 0.6931471805599453,
 'Jupiter': 0.6931471805599453,
 'planet': 0.6931471805599453,
 'is': 0.0,
 'the': 0.0,
 'from': 0.6931471805599453}
```

In [23]:
```
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])
df

```

Out[23]:

| | fourth | Sun | Planet | largest | Mars | Jupiter | planet | is | the | from |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.000000 | 0.000000 | 0.138629 | 0.138629 | 0.000000 | 0.138629 | 0.000000 | 0.0 | 0.0 | 0.000000 |
| **1** | 0.086643 | 0.086643 | 0.000000 | 0.000000 | 0.086643 | 0.000000 | 0.086643 | 0.0 | 0.0 | 0.086643 |

In [31]:
```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from nltk.corpus import stopwords
import nltk
import re
```

In [32]:
```
nltk.download('stopwords')
```

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

Out[32]:
```
True
```

In [33]:
```
stop_words = set(stopwords.words('english'))
```

In [37]:
```
data = {'Text': ["Agriculture is the backbone of our economy.",
                 "Farmers work hard to grow crops for food production.",
                 "Market trends affect crop prices and demand."]}
```

In [38]:
```
df = pd.DataFrame(data)
```

In [39]:
```
def clean_text(text):
    text = text.lower()  # Lowercase
    text = re.sub(r'\W+', ' ', text)  # Remove special characters
    text = ' '.join(word for word in text.split() if word not in stop_words)  # Remove stopword
    return text
```

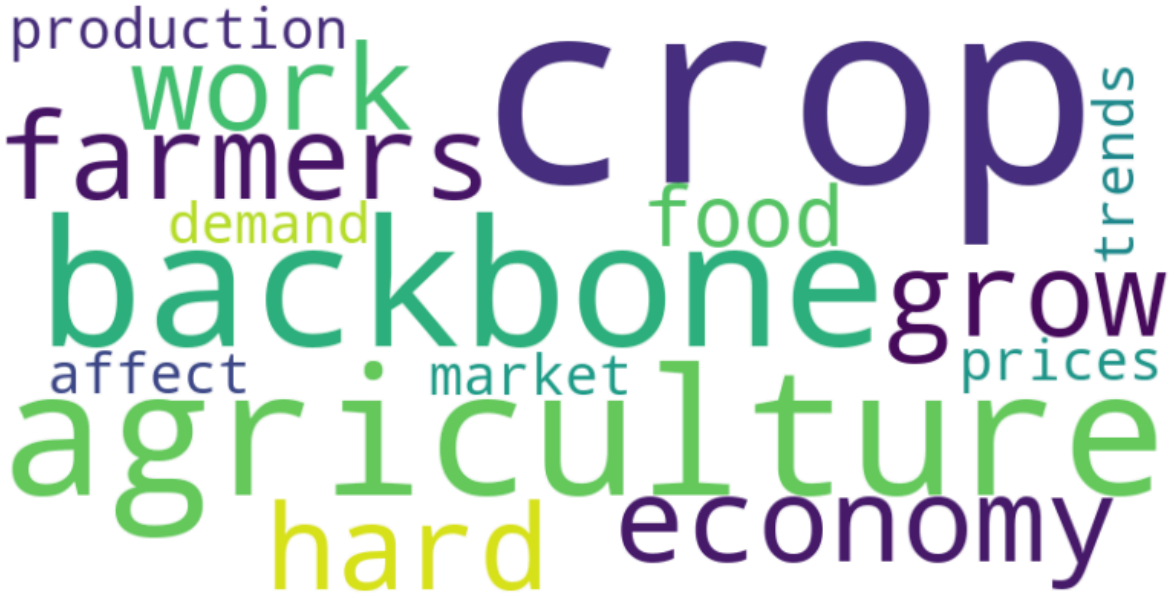
In [41]:
```
df['Cleaned_Text'] = df['Text'].apply(clean_text)
```

In [42]:
```
text = " ".join(df['Cleaned_Text'])
```

In [43]:
```
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)
```

In [44]:

```
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```




Name : Mohan Kadambande

Roll No. : 13212 (TECO-b1)