Group membes:

1. MEHJABIN RAHMAN -501200106
2. MAISHA LABIBA-501191497

PART1: Natural Language Processing- Finding Text Similarities

This part is about finding similarities in text and document. By considering following sentences and using any similarity or a clustering methods (e.g., k-Means or hierarchical clustering) show the more similar sentences. Use at least three methods from TF, TF-IDF, bag of the words Word2Vec or shingling techniques, and consequently, use a proper distance measure (e.g., Jaccard, Cosine, Edit and Hamming distance) cluster following data and show the similarity between them. Note that if you want to create your own distance measure, you should prove it is a valid distance measure.

First we import all the necessary libraries

In [1]:
```
import pandas as pd
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk import word_tokenize
from nltk.stem.porter import *
from nltk.stem.wordnet import WordNetLemmatizer
import re
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

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

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

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

Out[2]:
```
True
```

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

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

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

In [4]:
```
import nltk
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
```

```
[nltk_data]    Package omw-1.4 is already up-to-date!
```

Out[4]:
```
True
```

In [5]:
```
from sklearn.cluster import KMeans
```

In [6]:
```
stemmer=PorterStemmer()
lemmatizer=WordNetLemmatizer()
```

In [7]:
```
sentences= {
    "Id": [1, 2, 3, 4, 5,6],
    "Text": [
        "In the past John liked only sport but now he likes sport and politics",
        "Sam only liked politics but now he is fan of both music and politics",
        "Sara likes both books and politics but in the past she only read books",
        "Robert loved both books and nature but now he only reads books",
        "Linda liked books and sport but she only likes sport now",
        "Alison used to loved nature but currently she likes both nature and sport"
    ]
}
```

We have preprocessed the dataset. We have converted the sentences in lower case, remove stop words. Also applied stemming and lemmatizations.

In [8]:
```
def preprocessing(sentence):
    #converting to lowercase
    sentence=sentence.lower()
    #tokenization
    words=word_tokenize(sentence)
    #stemming
    words=[stemmer.stem(word) for word in words if word not in set(stopwords.words('engl
    #lemmatization
    words=[lemmatizer.lemmatize(word.lower()) for word in words]

    sentence=' '.join(words)
    return sentence
```

After applying preprocessing, our dataset looks like the following.

In [9]:
```
sentences['Text'] = [preprocessing(sentence) for sentence in sentences['Text']]

sentences
```

Out[9]:
```
{'Id': [1, 2, 3, 4, 5, 6],
 'Text': ['past john like sport like sport polit',
  'sam like polit fan music polit',
  'sara like book polit past read book',
  'robert love book natur read book',
  'linda like book sport like sport',
  'alison use love natur current like natur sport']}
```

In [19]:
```
# KMeans clustering is a method of clustering.
def clustering(matrix):
    number_of_clusters = 3
    km = KMeans(n_clusters=number_of_clusters)
    km.fit(matrix)


    # Print the cluster assignments for each sentence
    for i, sentence in enumerate(sentences['Text']):
        print(f"Sentence {i+1}: {sentences['Text'][i]} Cluster {km.labels_[i]+1}")
```

To find the similarities between sentences, we have used three methods (Bag-of-words, TF, and TF-IDF). Also we have also used the cosine similarity to measure the distance. And using K-means clustering to show the similar sentences.

In [20]:
```
# Function to apply bag of words model
def bow_similarity(data):
    vectorizer1 = CountVectorizer()
    vocab = vectorizer1.fit(data)
    X = vectorizer1.transform(data)
    ###
    print("A list of features:")
    print("====================")
    print(vocab.get_feature_names_out())

    print("BOW-weighted document-term matrix:")
    print("====================================")
    dp1=pd.DataFrame(X.toarray(),columns=vocab.get_feature_names_out())
    print(dp1)
    ###
    print("====================== \n")
    print("k means clustering \n")
    clustering(X)
    return cosine_similarity(X)

#importing TF using  parameter use_idf = False, norm = "l1"
def tf_similarity(data):
    vectorizer2 = TfidfVectorizer(use_idf = False, norm = "l1")
    tf_vocab = vectorizer2.fit(data)
    tf_matrix = vectorizer2.transform(data)
    ###
    print("A list of features:")
    print("====================")
    print(tf_vocab.get_feature_names_out())

    print("Tf-weighted document-term matrix:")
    print("====================================")
    dp=pd.DataFrame(tf_matrix.toarray(),columns=tf_vocab.get_feature_names_out())
    print(dp)
    ###
    print("====================== \n")
    print("k means clustering \n")
    clustering(tf_matrix)
    return cosine_similarity(tf_matrix)

# Importing TF-IDF using parameter norm = "l2"

def tfidf_similarity(data):
    vectorizer3 = TfidfVectorizer(norm = 'l2')
    tfidf_vocab = vectorizer3.fit(data)
    tfidf_matrix = vectorizer3.transform(data)
    ###
    print("A list of features:")
    print("====================")
    print(tfidf_vocab.get_feature_names_out())

    print("Tf-idf-weighted document-term matrix:")
    print("====================================")
    dp=pd.DataFrame(tfidf_matrix.toarray(),columns=tfidf_vocab.get_feature_names_out())
    print(dp)
    ###
    print("====================== \n")
    print("k means clustering \n")
    clustering(tfidf_matrix)
    return cosine_similarity(tfidf_matrix)




methods = {
    "TF with Cosine Similarity": tf_similarity,
```

```
    "Bag of words with Cosine Similarity": bow_similarity,
    "TF-IDF with Cosine Similarity": tfidf_similarity
}

# Apply the functions and print the similarity matrix and most similar sentences for eac
for method, function in methods.items():
    print('\n')
    print("====================================")
    print(f"--- {method} ---")
    similarity_matrix = function(sentences['Text'])
    print('\n')
    print("similarity matrix")
    print(similarity_matrix)
    print('\n')

    for i, row in enumerate(similarity_matrix):
        most_similar = sorted([(j, score) for j, score in enumerate(row) if j != i], key
        print(f"Most similar to sentence {i+1}:")
        for j, score in most_similar:
            print(f"\tSentence {j+1} with similarity score {score}")
    print()
```

```
====================================
--- TF with Cosine Similarity ---
A list of features:
====================
['alison' 'book' 'current' 'fan' 'john' 'like' 'linda' 'love' 'music'
 'natur' 'past' 'polit' 'read' 'robert' 'sam' 'sara' 'sport' 'use']
Tf-weighted document-term matrix:
====================================
   alison      book  current       fan      john      like     linda  \
0   0.000  0.000000    0.000  0.000000  0.142857  0.285714  0.000000
1   0.000  0.000000    0.000  0.166667  0.000000  0.166667  0.000000
2   0.000  0.285714    0.000  0.000000  0.000000  0.142857  0.000000
3   0.000  0.333333    0.000  0.000000  0.000000  0.000000  0.000000
4   0.000  0.166667    0.000  0.000000  0.000000  0.333333  0.166667
5   0.125  0.000000    0.125  0.000000  0.000000  0.125000  0.000000

       love     music     natur      past     polit      read    robert  \
0  0.000000  0.000000  0.000000  0.142857  0.142857  0.000000  0.000000
1  0.000000  0.166667  0.000000  0.000000  0.333333  0.000000  0.000000
2  0.000000  0.000000  0.000000  0.142857  0.142857  0.142857  0.000000
3  0.166667  0.000000  0.166667  0.000000  0.000000  0.166667  0.166667
4  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
5  0.125000  0.000000  0.250000  0.000000  0.000000  0.000000  0.000000

        sam      sara     sport    use
0  0.000000  0.000000  0.285714  0.000
1  0.166667  0.000000  0.000000  0.000
2  0.000000  0.142857  0.000000  0.000
3  0.000000  0.000000  0.000000  0.000
4  0.000000  0.000000  0.333333  0.000
5  0.000000  0.000000  0.125000  0.125
======================


k means clustering

Sentence 1: past john like sport like sport polit Cluster 1
Sentence 2: sam like polit fan music polit Cluster 3
Sentence 3: sara like book polit past read book Cluster 2
Sentence 4: robert love book natur read book Cluster 2
Sentence 5: linda like book sport like sport Cluster 1
Sentence 6: alison use love natur current like natur sport Cluster 1
```

```
similarity matrix
[[1.         0.42640143 0.40201513 0.         0.76277007 0.38138504]
 [0.42640143 1.         0.35355339 0.         0.2236068  0.1118034 ]
 [0.40201513 0.35355339 1.         0.58925565 0.42163702 0.10540926]
 [0.         0.         0.58925565 1.         0.2236068  0.3354102 ]
 [0.76277007 0.2236068  0.42163702 0.2236068  1.         0.4       ]
 [0.38138504 0.1118034  0.10540926 0.3354102  0.4        1.        ]]


Most similar to sentence 1:
        Sentence 5 with similarity score 0.7627700713964738
Most similar to sentence 2:
        Sentence 1 with similarity score 0.42640143271122083
Most similar to sentence 3:
        Sentence 4 with similarity score 0.5892556509887895
Most similar to sentence 4:
        Sentence 3 with similarity score 0.5892556509887895
Most similar to sentence 5:
        Sentence 1 with similarity score 0.7627700713964738
Most similar to sentence 6:
        Sentence 5 with similarity score 0.39999999999999997



====================================
--- Bag of words with Cosine Similarity ---
A list of features:
====================
['alison' 'book' 'current' 'fan' 'john' 'like' 'linda' 'love' 'music'
 'natur' 'past' 'polit' 'read' 'robert' 'sam' 'sara' 'sport' 'use']
BOW-weighted document-term matrix:
====================================
   alison  book  current  fan  john  like  linda  love  music  natur  past  \
0       0     0        0    0     1     2      0     0      0      0     1
1       0     0        0    1     0     1      0     0      1      0     0
2       0     2        0    0     0     1      0     0      0      0     1
3       0     2        0    0     0     0      0     1      0      1     0
4       0     1        0    0     0     2      1     0      0      0     0
5       1     0        1    0     0     1      0     1      0      2     0

   polit  read  robert  sam  sara  sport  use
0      1     0       0    0     0      2    0
1      2     0       0    1     0      0    0
2      1     1       0    0     1      0    0
3      0     1       1    0     0      0    0
4      0     0       0    0     0      2    0
5      0     0       0    0     0      1    1
======================

k means clustering

Sentence 1: past john like sport like sport polit Cluster 1
Sentence 2: sam like polit fan music polit Cluster 1
Sentence 3: sara like book polit past read book Cluster 2
Sentence 4: robert love book natur read book Cluster 2
Sentence 5: linda like book sport like sport Cluster 1
Sentence 6: alison use love natur current like natur sport Cluster 3


similarity matrix
[[1.         0.42640143 0.40201513 0.         0.76277007 0.38138504]
 [0.42640143 1.         0.35355339 0.         0.2236068  0.1118034 ]
 [0.40201513 0.35355339 1.         0.58925565 0.42163702 0.10540926]
 [0.         0.         0.58925565 1.         0.2236068  0.3354102 ]
 [0.76277007 0.2236068  0.42163702 0.2236068  1.         0.4       ]
 [0.38138504 0.1118034  0.10540926 0.3354102  0.4        1.        ]]
```

```
Most similar to sentence 1:
       Sentence 5 with similarity score 0.7627700713964739
Most similar to sentence 2:
       Sentence 1 with similarity score 0.42640143271122083
Most similar to sentence 3:
       Sentence 4 with similarity score 0.5892556509887895
Most similar to sentence 4:
       Sentence 3 with similarity score 0.5892556509887895
Most similar to sentence 5:
       Sentence 1 with similarity score 0.7627700713964739
Most similar to sentence 6:
       Sentence 5 with similarity score 0.4



==================================
--- TF-IDF with Cosine Similarity ---
A list of features:
====================
['alison' 'book' 'current' 'fan' 'john' 'like' 'linda' 'love' 'music'
 'natur' 'past' 'polit' 'read' 'robert' 'sam' 'sara' 'sport' 'use']
Tf-idf-weighted document-term matrix:
====================================
    alison      book  current       fan      john      like     linda  \
0  0.00000  0.000000  0.00000  0.000000  0.441993  0.452889  0.000000
1  0.00000  0.000000  0.00000  0.439389  0.000000  0.225111  0.000000
2  0.00000  0.618987  0.00000  0.000000  0.000000  0.229032  0.000000
3  0.00000  0.623322  0.00000  0.000000  0.000000  0.000000  0.000000
4  0.00000  0.328320  0.00000  0.000000  0.000000  0.485929  0.474237
5  0.37519  0.000000  0.37519  0.000000  0.000000  0.192220  0.000000

       love     music     natur      past     polit      read    robert  \
0  0.000000  0.000000  0.000000  0.362440  0.305997  0.000000  0.000000
1  0.000000  0.439389  0.000000  0.000000  0.608389  0.000000  0.000000
2  0.000000  0.000000  0.000000  0.366582  0.309494  0.366582  0.000000
3  0.369149  0.000000  0.369149  0.000000  0.000000  0.369149  0.450174
4  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
5  0.307662  0.000000  0.615323  0.000000  0.000000  0.000000  0.000000

        sam      sara     sport      use
0  0.000000  0.000000  0.611994  0.00000
1  0.439389  0.000000  0.000000  0.00000
2  0.000000  0.447043  0.000000  0.00000
3  0.000000  0.000000  0.000000  0.00000
4  0.000000  0.000000  0.656641  0.00000
5  0.000000  0.000000  0.259749  0.37519
======================

k means clustering

Sentence 1: past john like sport like sport polit Cluster 1
Sentence 2: sam like polit fan music polit Cluster 3
Sentence 3: sara like book polit past read book Cluster 2
Sentence 4: robert love book natur read book Cluster 2
Sentence 5: linda like book sport like sport Cluster 1
Sentence 6: alison use love natur current like natur sport Cluster 1


similarity matrix
[[1.         0.28811542 0.33129449 0.         0.62193243 0.24601928]
 [0.28811542 1.         0.23985016 0.         0.10938785 0.04327081]
 [0.33129449 0.23985016 1.         0.52115186 0.31451969 0.04402461]
 [0.         0.         0.52115186 1.         0.20464939 0.34071893]
 [0.62193243 0.10938785 0.31451969 0.20464939 1.         0.2639672 ]
```

```
 [0.24601928 0.04327081 0.04402461 0.34071893 0.2639672  1.        ]]


Most similar to sentence 1:
        Sentence 5 with similarity score 0.6219324252261071
Most similar to sentence 2:
        Sentence 1 with similarity score 0.28811542442208304
Most similar to sentence 3:
        Sentence 4 with similarity score 0.5211518578703335
Most similar to sentence 4:
        Sentence 3 with similarity score 0.5211518578703335
Most similar to sentence 5:
        Sentence 1 with similarity score 0.6219324252261071
Most similar to sentence 6:
        Sentence 4 with similarity score 0.3407189331492442
```

In [ ]: