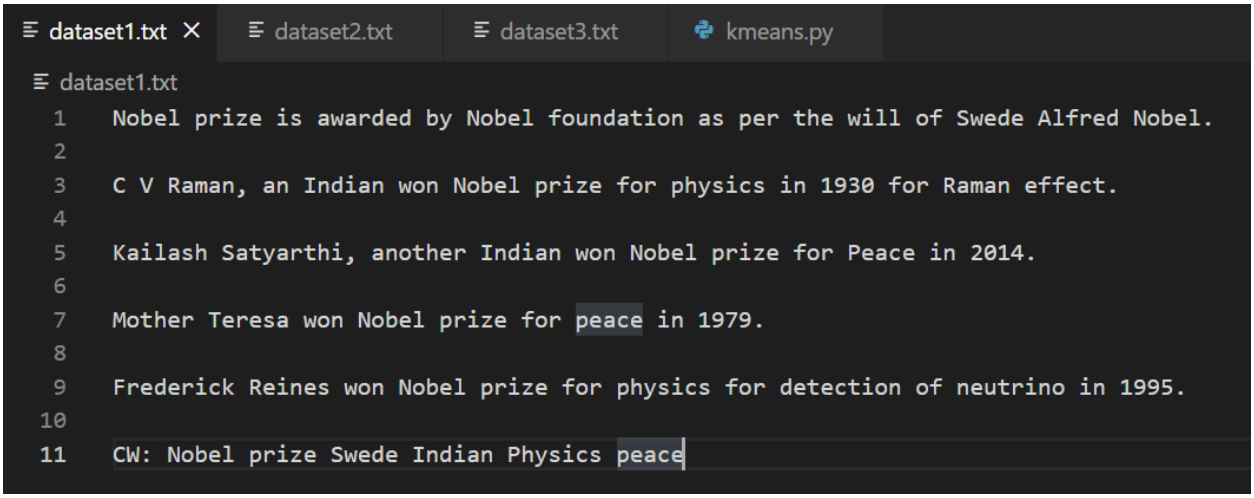
Gerosh Shibu George

19BCE1403

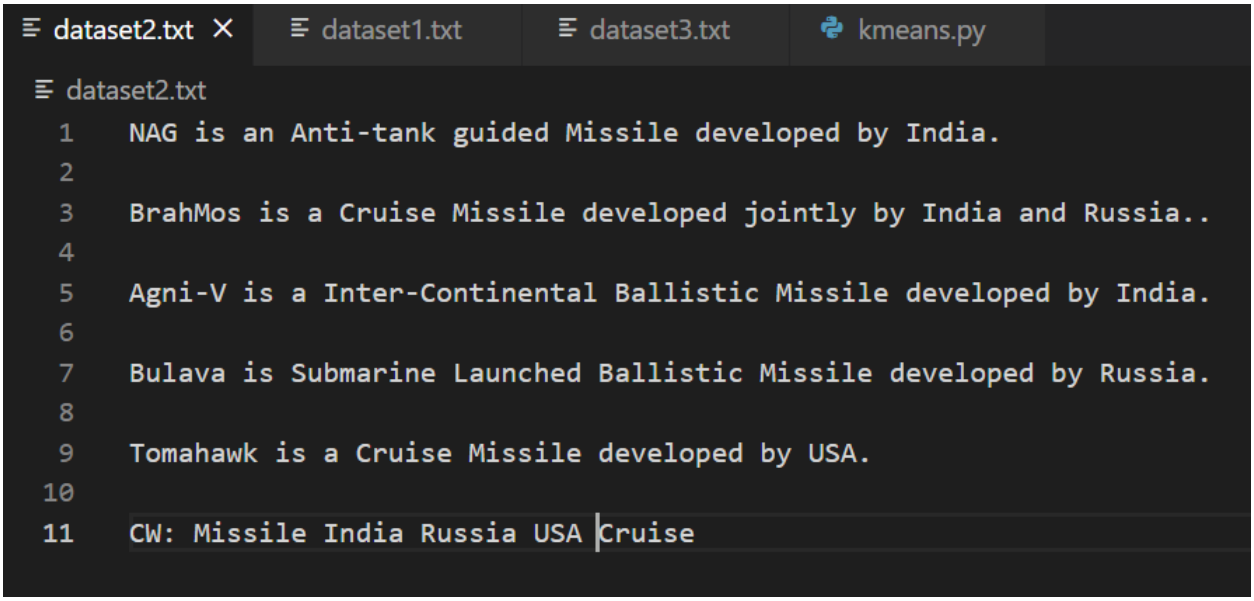
WM Lab-8

K-means clustering lab assignment

Dataset 1



Dataset 2



Dataset 3

Question:

NOTE: [1] Input the datasetx from datasetM.txt file and read [M=1,2,3…]
[2] Read the content words of the dataset from content_wordsM.txt [M=1,2,3]
[2] Write a Construct_VSM_data_point routine to translate document to a Vector space Model data point
[4] Write a Euclid_Dist() to calculate distance between two documents and print the euclid_dist_matrix out and also store it in a two dimensional array or a list.

Read K from the user

Write a K-mean clustering routine that clusters all the following datasets with the given Content words.

Give K=2,3,4 and print the output.

Program Code:

```
import numpy as np
from random import sample
import random
random.seed(19)

datasets = {}

def load_dataset():

    for i in range(3):
        datasets[i+1]={}
        filename = "dataset" + str((i+1)) + ".txt"
        documents = []
        content_words = []

        with open(filename,"r") as fp:

            for line in fp.readlines():
                if len(line)> 1:

                    if not line.startswith('CW:'):
                        documents.append((line.replace(".\n","")).lower())
                    else:
                        content_words = line.split()
                        content_words = content_words[1:]
                        content_words = [ w.lower() for w in content_words]

        documents_vsm = []

        for doc in documents:
            vsm = VSM(doc,content_words)
            documents_vsm.append(vsm)

        datasets[i+1]['VSM'] = documents_vsm
        datasets[i+1]['cluster'] = [0] * len(documents)

    return datasets


def VSM(document,content_words):
    vsm = []
    for word in content_words:
```

```
        vsm.append(document.count(word))
    return vsm

def Euclid_Dist(VSM):

    dist_mat = []
    dist_row = []

    for i in range(len(VSM)):

        x = np.array(VSM[i])
        dist_row = [0] * (i+1)

        for j in range(i+1,len(VSM)):

            y = np.array(VSM[j])
            dist = np.round(np.linalg.norm(x-y),3)
            dist_row.append(dist)
        dist_mat.append(dist_row)

    return np.matrix(dist_mat)



def calculate_new_centroids(VSM,K,clusters):

    unique = [ i for i in range(K) ]
    centroids = []

    for cid in unique:
        centroid = np.zeros(len(VSM[0]))
        for index,c in enumerate(clusters):
            if cid == c:
                point = np.array(VSM[index], dtype='int64')
                centroid = centroid + point
        count = clusters.count(cid)
        centroid = centroid / count
        centroids.append(centroid)

    return centroids

def list_equal(l1,l2):

    for i in range(len(l1)):
        if l1[i]!=l2[i]:
```

```
            return False

    return True

def Kmeans(VSM,K):

    centroids = sample(VSM,K)

    # print("[INFO] Initial Centroids:")
    # for row in centroids:
    #     print(row)

    final_clusters = [-1]*len(VSM)

    #print(f"[INFO] Initial Clusters: {final_clusters}")

    while(True):

        clusters = []

        for i,vec in enumerate(VSM):

            distances=[]

            for centroid in centroids:
                x = np.array(vec)
                y = np.array(centroid)
                dist = np.round(np.linalg.norm(x-y),3)
                distances.append(dist)

            #print(f"[INFO] Doc{i+1} Distance centroid vector: {distances}")
            clusters.append(np.argmin(distances,axis=0))


        #print(f"[INFO] Clusters formed: {clusters}")

        if list_equal(final_clusters,clusters):
            break

        final_clusters = clusters

        #print("\nNext Iterations\n")

        # calculate new centroids
        #unique = set(clusters)
```

```
        centroids = []

        centroids = calculate_new_centroids(VSM,K, clusters)

        #print("[INFO] New Centroids:")
        # for row in centroids:
        #     print(row)

    return final_clusters


def init(K=2):

    if K==1:
        K=2

    datasets = load_dataset()

    #Kmeans(datasets[1]['VSM'],K)

    for doc in datasets:

        print(f"\nVSM of dataset No: {doc}: \n")
        for row in datasets[doc]['VSM']:
            print(row)
        print("\n")

        print("Distance Matrix:\n")
        m = Euclid_Dist(datasets[doc]['VSM'])
        print(m)
        print("\n")

        print(f"Result of Kmeans cluster with K ({K}): ",end="")
        clusters = Kmeans(datasets[doc]['VSM'],K)
        print(clusters)
        print("\n")


K = int(input('Enter the value of K: '))
print()
init(K)
```

OUTPUT:

For K = 2

```
(venv) PS C:\Users\Gerosh\Desktop\VIT\Third Year\Web Minning\Programs> python kmeans.py
Enter the value of K: 2


VSM of dataset No: 1:

[3, 1, 1, 0, 0, 0]
[1, 1, 0, 1, 1, 0]
[1, 1, 0, 1, 0, 1]
[1, 1, 0, 0, 0, 1]
[1, 1, 0, 0, 1, 0]


Distance Matrix:

[[0.    2.646 2.646 2.449 2.449]
 [0.    0.    1.414 1.732 1.   ]
 [0.    0.    0.    1.    1.732]
 [0.    0.    0.    0.    1.414]
 [0.    0.    0.    0.    0.   ]]


Result of Kmeans cluster with K (2): [0, 1, 1, 1, 1]
```

```
VSM of dataset No: 2:

[1, 1, 0, 0, 0]
[1, 1, 1, 0, 1]
[1, 1, 0, 0, 0]
[1, 0, 1, 0, 0]
[1, 0, 0, 1, 1]


Distance Matrix:

[[0.    1.414 0.    1.414 1.732]
 [0.    0.    1.414 1.414 1.732]
 [0.    0.    0.    1.414 1.732]
 [0.    0.    0.    0.    1.732]
 [0.    0.    0.    0.    0.   ]]


Result of Kmeans cluster with K (2): [1, 1, 1, 1, 0]
```

```
VSM of dataset No: 3:

[1, 1, 0, 0]
[1, 1, 0, 0]
[1, 0, 1, 0]
[1, 0, 1, 1]
[0, 1, 1, 0]


Distance Matrix:

[[0.    0.    1.414 1.732 1.414]
 [0.    0.    1.414 1.732 1.414]
 [0.    0.    0.    1.    1.414]
 [0.    0.    0.    0.    1.732]
 [0.    0.    0.    0.    0.   ]]


Result of Kmeans cluster with K (2): [1, 1, 1, 0, 1]
```

For K = 3

```
(venv) PS C:\Users\Gerosh\Desktop\VIT\Third Year\Web Minning\Programs> python kmeans.py
Enter the value of K: 3


VSM of dataset No: 1:

[3, 1, 1, 0, 0, 0]
[1, 1, 0, 1, 1, 0]
[1, 1, 0, 1, 0, 1]
[1, 1, 0, 0, 0, 1]
[1, 1, 0, 0, 1, 0]


Distance Matrix:

[[0.    2.646 2.646 2.449 2.449]
 [0.    0.    1.414 1.732 1.   ]
 [0.    0.    0.    1.    1.732]
 [0.    0.    0.    0.    1.414]
 [0.    0.    0.    0.    0.   ]]


Result of Kmeans cluster with K (3): [0, 1, 2, 2, 1]
```

```
VSM of dataset No: 2:

[1, 1, 0, 0, 0]
[1, 1, 1, 0, 1]
[1, 1, 0, 0, 0]
[1, 0, 1, 0, 0]
[1, 0, 0, 1, 1]


Distance Matrix:

[[0.    1.414 0.    1.414 1.732]
 [0.    0.    1.414 1.414 1.732]
 [0.    0.    0.    1.414 1.732]
 [0.    0.    0.    0.    1.732]
 [0.    0.    0.    0.    0.   ]]


Result of Kmeans cluster with K (3): [0, 0, 0, 1, 2]
```

```
VSM of dataset No: 3:

[1, 1, 0, 0]
[1, 1, 0, 0]
[1, 0, 1, 0]
[1, 0, 1, 1]
[0, 1, 1, 0]


Distance Matrix:

[[0.    0.    1.414 1.732 1.414]
 [0.    0.    1.414 1.732 1.414]
 [0.    0.    0.    1.    1.414]
 [0.    0.    0.    0.    1.732]
 [0.    0.    0.    0.    0.   ]]


Result of Kmeans cluster with K (3): [0, 0, 1, 2, 0]
```

For K = 4

```
(venv) PS C:\Users\Gerosh\Desktop\VIT\Third Year\Web Minning\Programs> python kmeans.py
Enter the value of K: 4


VSM of dataset No: 1:

[3, 1, 1, 0, 0, 0]
[1, 1, 0, 1, 1, 0]
[1, 1, 0, 1, 0, 1]
[1, 1, 0, 0, 0, 1]
[1, 1, 0, 0, 1, 0]


Distance Matrix:

[[0.    2.646 2.646 2.449 2.449]
 [0.    0.    1.414 1.732 1.   ]
 [0.    0.    0.    1.    1.732]
 [0.    0.    0.    0.    1.414]
 [0.    0.    0.    0.    0.   ]]


Result of Kmeans cluster with K (4): [0, 1, 2, 3, 1]
```

```
VSM of dataset No: 2:

[1, 1, 0, 0, 0]
[1, 1, 1, 0, 1]
[1, 1, 0, 0, 0]
[1, 0, 1, 0, 0]
[1, 0, 0, 1, 1]


Distance Matrix:

[[0.    1.414 0.    1.414 1.732]
 [0.    0.    1.414 1.414 1.732]
 [0.    0.    0.    1.414 1.732]
 [0.    0.    0.    0.    1.732]
 [0.    0.    0.    0.    0.   ]]


Result of Kmeans cluster with K (4): [1, 3, 1, 0, 2]
```

```
VSM of dataset No: 3:

[1, 1, 0, 0]
[1, 1, 0, 0]
[1, 0, 1, 0]
[1, 0, 1, 1]
[0, 1, 1, 0]


Distance Matrix:

[[0.    0.    1.414 1.732 1.414]
 [0.    0.    1.414 1.732 1.414]
 [0.    0.    0.    1.    1.414]
 [0.    0.    0.    0.    1.732]
 [0.    0.    0.    0.    0.   ]]


Result of Kmeans cluster with K (4): [1, 1, 2, 3, 0]
```