WEB MINING
LAB

Faculty: Dr.Sridhar.R

LAB7
DATE: 7TH OCT 2021

VIT CHENNAI

BELIDA KARTHIK
19BCE1446

Aim: To understand and implement k-means clustering

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.

DATA SET 1:

Doc 1: Nobel prize is awarded by Nobel foundation as per the will of Swede Alfred Nobel
Doc-2 C V Raman, an Indian won Nobel prize for physics in 1930 for Raman effect.
Doc-3 Kailash Satyarthi, another Indian won Nobel prize for Peace in 2014
Doc-4 Mother Teresa won Nobel prize for peace in 1979.
Doc-5 Frederick Reines won Nobel prize for physics for detection of neutrino in 1995.

[1] Use the following content words to represent the above document in vector space model. Frequency of the word is the magnitude in the dimension of the content word. Ignore first letter capitalization while counting frequency. [a] Nobel [b] prize [c] Swede [d] Indian [e] Physics [f] peace
[2] Use Euclidean distance L2 to find th

Code:

```
import string
import numpy as np
import os
import math

class document_clustering(object):

    def __init__(self, file_dict, word_list, k):
        self.file_dict = file_dict
        self.word_list = word_list
        self.k = k

    def tokenize_document(self, document):

        terms = document.lower().split()
        return [term.strip(string.punctuation) for term in terms]

    def create_word_listing(self):

        self.listing_dict_ = {}
        dir = os.path.dirname(__file__)
        for id in self.file_dict:
            temp_word_list = []
            filename=os.path.join(dir, self.file_dict[id])
            f = open(filename, 'r')
            document = f.read()
            terms = self.tokenize_document(document)
            for term in self.word_list:
                temp_word_list.append(terms.count(term.lower()))
            self.listing_dict_[id] = temp_word_list

        print('Word listing of each document')
        for id in self.listing_dict_:
            print('%d\t%s' % (id, self.listing_dict_[id]))
```

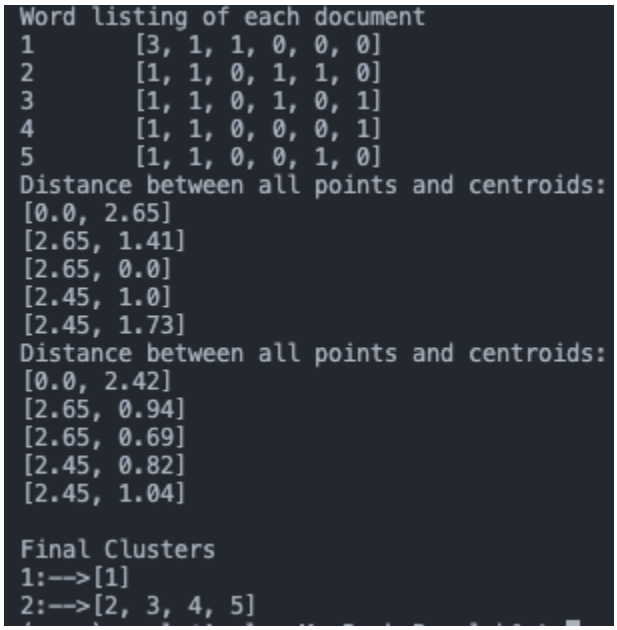
```
    def find_centroid(self, feature):

        distances = []
        for centroid in self.centroids_:
            dist = 0
            for i in range(0,len(self.centroids_[centroid])):
                dist += pow((self.centroids_[centroid][i] - feature[i]),2)
            dist = math.sqrt(dist)
            distances.append(round(dist,2))

        print("Distance between all points and centroids: ")
        print(distances)
        return np.argmin(distances)

    def kmeans_clustering(self):

        centroid = [1,5,8]
        self.centroids_ = {}
        for i in range(self.k):
            self.centroids_[i] = self.listing_dict_[centroid[i]]

        for i in range(2):
            self.classes_ = {}
            self.features_ = {}

            for i in range(self.k):
                self.classes_[i] = []
                self.features_[i] = [self.centroids_[i]]

            for id in self.listing_dict_:
                classification = self.find_centroid(self.listing_dict_[id])
                self.classes_[classification].append(id)
                self.features_[classification].append(self.listing_dict_[id])

            previous = dict(self.centroids_)
```

```
            for i in self.features_:
                self.centroids_[i] = np.average(self.features_[i], axis = 0)

            isOptimal = True

            for centroid in self.centroids_:
                original_centroid = np.array(previous[centroid])
                curr_centroid = self.centroids_[centroid]

                if np.sum(original_centroid - curr_centroid) != 0:
                    isOptimal = False

            # Breaking the results if the centroids found are optimal
            if isOptimal:
                break

    def print_clusters(self):
        print('\nFinal Clusters')
        for i in self.classes_:
            print('%d:-->%s' % (i+1, self.classes_[i]))



file_dict = {1: 'documents/doc1.txt',
             2: 'documents/doc2.txt',
             3: 'documents/doc3.txt',
             4: 'documents/doc4.txt',
             5: 'documents/doc5.txt',
             6: 'documents/doc6.txt',
             7: 'documents/doc7.txt',
             8: 'documents/doc8.txt',
             9: 'documents/doc9.txt'}

word_list = ['ISRO','moon', 'water', 'mars', 'DRDO', 'missile', 'fighter' ,'IAF']
```

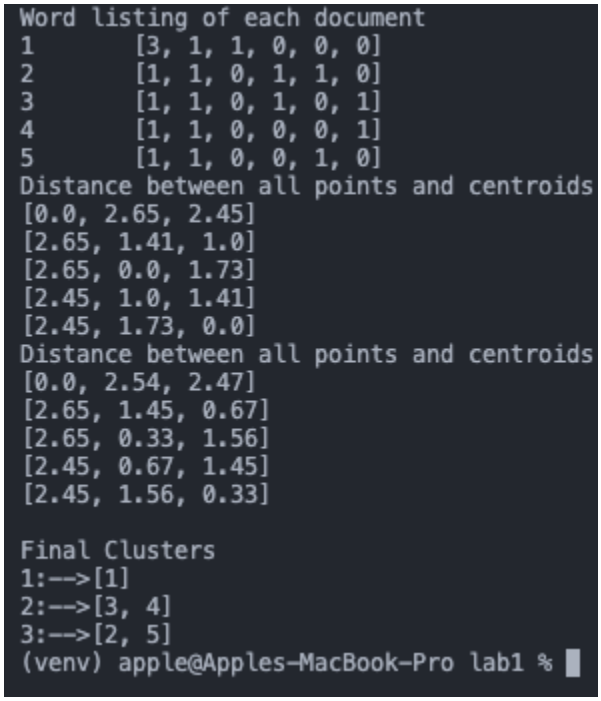
```
document_cluster = document_clustering(file_dict = file_dict, word_list = word_list, k = 3)
document_cluster.create_word_listing()
document_cluster.kmeans_clustering()
document_cluster.print_clusters()
```

OutPut:

For k=2



For k=3



For k=4

```
Word listing of each document
1       [3, 1, 1, 0, 0, 0]
2       [1, 1, 0, 1, 1, 0]
3       [1, 1, 0, 1, 0, 1]
4       [1, 1, 0, 0, 0, 1]
5       [1, 1, 0, 0, 1, 0]
Distance between all points and centroids:
[0.0, 2.65, 2.45, 2.45]
[2.65, 1.41, 1.0, 1.73]
[2.65, 0.0, 1.73, 1.0]
[2.45, 1.0, 1.41, 0.0]
[2.45, 1.73, 0.0, 1.41]
Distance between all points and centroids:
[0.0, 2.65, 2.47, 2.45]
[2.65, 1.41, 0.67, 1.73]
[2.65, 0.0, 1.56, 1.0]
[2.45, 1.0, 1.45, 0.0]
[2.45, 1.73, 0.33, 1.41]

Final Clusters
1:-->[1]
2:-->[3]
3:-->[2, 5]
4:-->[4]
```

DataSet-2

k=2

```
Word listing of each document
1       [1, 1, 0, 0, 0]
2       [1, 1, 1, 0, 1]
3       [1, 1, 0, 0, 0]
4       [1, 0, 1, 0, 0]
5       [1, 0, 0, 1, 1]
Distance between all points and centroids:
[0.0, 0.0]
[1.41, 1.41]
[0.0, 0.0]
[1.41, 1.41]
[1.73, 1.73]
Distance between all points and centroids:
[0.6, 0.0]
[1.01, 1.41]
[0.6, 0.0]
[1.01, 1.41]
[1.3, 1.73]

Final Clusters
1:-->[2, 4, 5]
2:-->[1, 3]
```

k=3

```
Word listing of each document
1       [1, 1, 0, 0, 0]
2       [1, 1, 1, 0, 1]
3       [1, 1, 0, 0, 0]
4       [1, 0, 1, 0, 0]
5       [1, 0, 0, 1, 1]
Distance between all points and centroids:
[0.0, 0.0, 1.73]
[1.41, 1.41, 1.73]
[0.0, 0.0, 1.73]
[1.41, 1.41, 1.73]
[1.73, 1.73, 0.0]
Distance between all points and centroids:
[0.49, 0.0, 1.73]
[1.02, 1.41, 1.73]
[0.49, 0.0, 1.73]
[1.02, 1.41, 1.73]
[1.56, 1.73, 0.0]

Final Clusters
1:-->[2, 4]
2:-->[1, 3]
3:-->[5]
```

k=4

```
Word listing of each document
1       [1, 1, 0, 0, 0]
2       [1, 1, 1, 0, 1]
3       [1, 1, 0, 0, 0]
4       [1, 0, 1, 0, 0]
5       [1, 0, 0, 1, 1]
Distance between all points and centroids:
[0.0, 0.0, 1.73, 1.41]
[1.41, 1.41, 1.73, 1.41]
[0.0, 0.0, 1.73, 1.41]
[1.41, 1.41, 1.73, 0.0]
[1.73, 1.73, 0.0, 1.73]
Distance between all points and centroids:
[0.35, 0.0, 1.73, 1.41]
[1.06, 1.41, 1.73, 1.41]
[0.35, 0.0, 1.73, 1.41]
[1.27, 1.41, 1.73, 0.0]
[1.62, 1.73, 0.0, 1.73]

Final Clusters
1:-->[2]
2:-->[1, 3]
3:-->[5]
4:-->[4]
```

DataSet -3

For k=2

```
Word listing of each document
1       [1, 0, 0, 0]
2       [1, 1, 0, 0]
3       [1, 0, 1, 0]
4       [1, 0, 1, 1]
5       [0, 1, 1, 0]
Distance between all points and centroids:
[0.0, 1.0]
[1.0, 1.41]
[1.0, 0.0]
[1.41, 1.0]
[1.73, 1.41]
Distance between all points and centroids:
[0.33, 1.09]
[0.67, 1.3]
[1.05, 0.43]
[1.45, 0.83]
[1.56, 1.09]

Final Clusters
1:-->[1, 2]
2:-->[3, 4, 5]
```

For k=3

```
Word listing of each document
1       [1, 0, 0, 0]
2       [1, 1, 0, 0]
3       [1, 0, 1, 0]
4       [1, 0, 1, 1]
5       [0, 1, 1, 0]
Distance between all points and centroids:
[0.0, 1.0, 1.73]
[1.0, 1.41, 1.41]
[1.0, 0.0, 1.41]
[1.41, 1.0, 1.73]
[1.73, 1.41, 0.0]
Distance between all points and centroids:
[0.33, 1.05, 1.73]
[0.67, 1.45, 1.41]
[1.05, 0.33, 1.41]
[1.45, 0.67, 1.73]
[1.56, 1.45, 0.0]

Final Clusters
1:-->[1, 2]
2:-->[3, 4]
3:-->[5]
```

For k=4

```
Word listing of each document
1       [1, 0, 0, 0]
2       [1, 1, 0, 0]
3       [1, 0, 1, 0]
4       [1, 0, 1, 1]
5       [0, 1, 1, 0]
Distance between all points and centroids:
[0.0, 1.0, 1.73, 1.41]
[1.0, 1.41, 1.41, 1.73]
[1.0, 0.0, 1.41, 1.0]
[1.41, 1.0, 1.73, 0.0]
[1.73, 1.41, 0.0, 1.73]
Distance between all points and centroids:
[0.33, 1.0, 1.73, 1.41]
[0.67, 1.41, 1.41, 1.73]
[1.05, 0.0, 1.41, 1.0]
[1.45, 1.0, 1.73, 0.0]
[1.56, 1.41, 0.0, 1.73]

Final Clusters
1:-->[1, 2]
2:-->[3]
3:-->[5]
4:-->[4]
```