CSE 601 – Data Mining and Bioinformatics

Project 2: Clustering Algorithms

Team members:

Kiran Ketan Nevrekar, 50336915, kiranket

Revathy Narayanan, 50336857, revathyn

Saranya Saravanan, 50314931, saravan2

Objective: The aim of this project is to implement the given five clustering algorithms such as K-Means clustering, Hierarchical clustering with minimum approach, density, mixture and spectral clustering to group similar data points.

Clustering: Clustering is an unsupervised machine learning technique that groups data points into a group that has similar values than the ones in other clusters. The goal of the clustering algorithm is to achieve high inter-clustering similarity and low intra-cluster similarity. There are various types of clustering methods that can be performed depending on the type of data provided and some of them are K-means, density based, hierarchical, mixture and spectral model clustering algorithms.

Evaluation methods: There are many ways to evaluate clustering algorithms. Rand Index and Jaccard coefficient are two methods used to compare the clustering results to the ground truth cluster.

Jaccard Coefficient: Jaccard coefficient measures the similarity between the given sample sets.The formula to compute Jaccard coefficient are as follows.

Jaccard coefficient = (TruePositive) / (TruePositive + FalsePositive + FalseNegative)

Rand Index: Rand Index is used to measure similarity between two data clusterings. The formula to compute the Rand Index are as follows.

Rand Index = (TruePositive + TrueNegative) / (TruePositive + FalsePositive + TrueNegative + FalseNegative)

1. K-means Clustering Algorithm

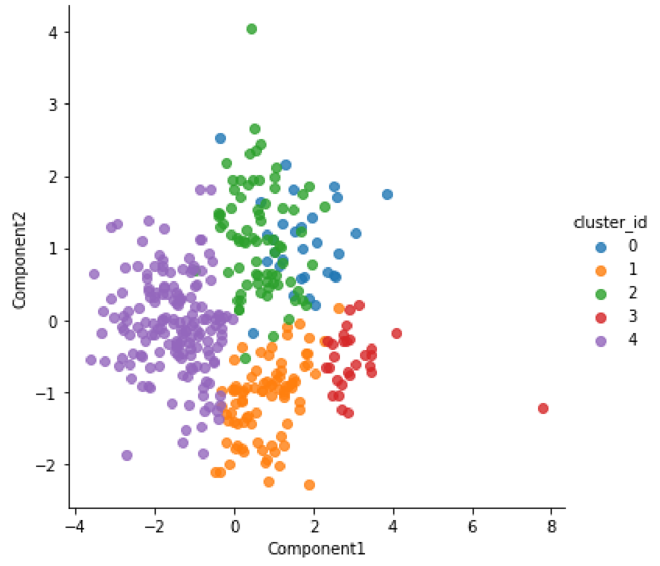
K-means Clustering: K-means clustering is an unsupervised machine learning clustering algorithm that uses centroid and distances to group similar data points in one cluster. The number of centroids is dependent on the k value and given the data, the algorithm tries to minimize the distance between the data points and the centroids.

Implementation:

1. A class named 'kmeansclustering' is defined under which two other methods are defined.
2. The method '_init_' has parameters self, k, maxiterationnumber and centroid which are passed to the next function with 'self' as reference.
3. The method 'clustering' contains all the functionalities starting from starting the iteration till final clusters are formed.
4. There are three loops, dedicated for iterating till a particular number, iterating through rows of data and also centroids.
5. In each iteration, minimum euclidean distance is considered and the data points get clustered accordingly.
6. For every clustering that happens, centroid is updated by taking mean.
7. The iteration stops when the centroids do not change and the final cluster is formed.

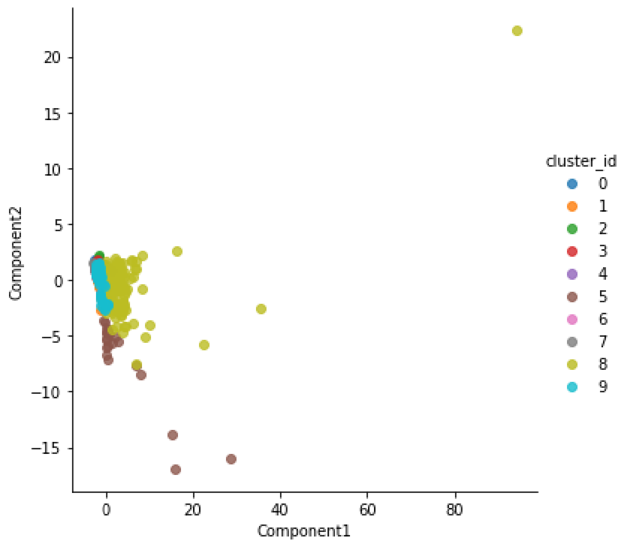
Packages used: Pandas, numpy, matplotlib

Result:
Cho.txt: k=5



```
Jaccard Coefficient =  0.41674057649667406
Rand Index =  0.7881419635426454
```

```
Iyer.txt: k =10
```



```
Jaccard Coefficient =  0.22227782941302637
Rand Index =  0.7965123892116772
```

<u>2. Hierarchical Agglomerative clustering with Min approach</u>

Hierarchical Clustering: Hierarchical clustering is an unsupervised machine learning clustering algorithm that groups data points with similar features or attributes. There are three approaches to this clustering algorithm and they are as follows: Minimum, maximum and average approach. Minimum approach is the distance between the closest members of its two clusters. Maximum approach is the distance between the members that are farthest apart. Average approach involves looking at the distances between all pairs and averages all of these distances. This method involves looking at the distances between all pairs and averages all of these distances. This is also called Unweighted Pair Group Mean Averaging. We have used minimum approach for our implementation.

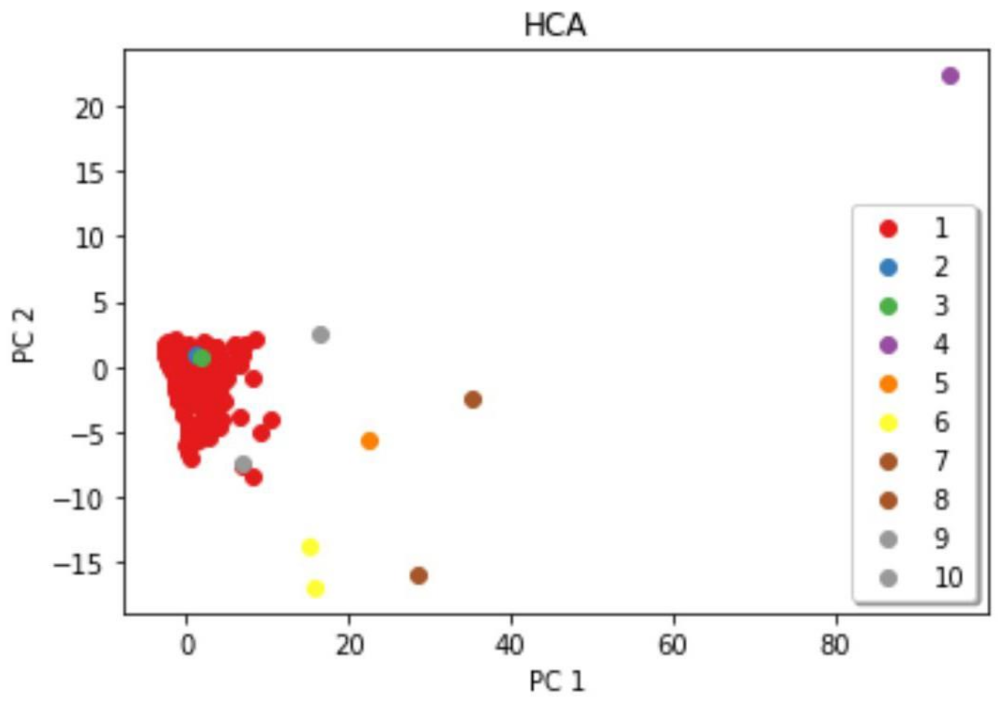
Implementation:

1. The data file and number of clusters is obtained from the user.
2. Pre processing of the data is done by removing unwanted columns like cluster id.etc. Groundtruth column is stored separately before dropping it from data.

3. Initially, distance matrix for the data points is computed using an in-built numpy function distance_matrix which computes distance of one point with other points and stores it as a matrix.
4. All the data points in this matrix are considered as individual clusters initially.
5. Then minimum non zero distance is found from matrix using np.min(distanceMatrix.nonzero) and its corresponding data points are found.
6. Also the indexes of these data points is found using np.where function.
7. Now since the distance is minimum,those points belong to one cluster so they are merged and distance matrix is to be updated.
8. This is done using updateCluster method by performing the above process by looping through all the dasata points in cluster's range.
9. Output is a list of final cluster values.
10. Next, plot these clusters by using the PCA() method from matplotlibrary from sklearn.
11. Compute Jaccard coefficient and Rand index by using its corresponding formula and display them.

Packages used: Pandas, numpy, matplotlib
Pros: 1) Does not need any particular number of clusters
2) Resistant to noise
Cons: 1) Once the merging of clusters is done, it cannot be undone.

Result:iyer.txt

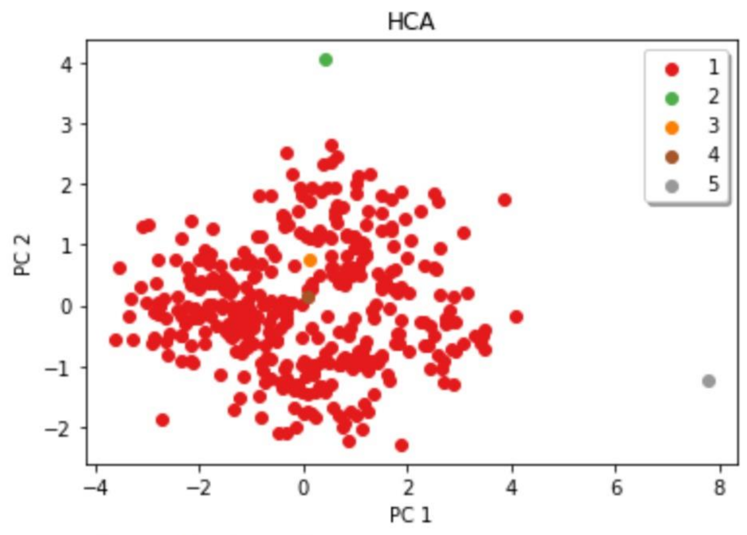


```
Jaccard Coefficient is  0.15824309696642858
Rand Index is  0.1882868355974245
```

Jaccard coefficient: 0.1582
Rand Index : 0.1882

Cho.txt

```
*c* argument looks like a single numeric RGB or RGBA s
```



```
Jaccard Coefficient is  0.2297361848480299
Rand Index is  0.24242261537222476
```

3. Density-based

Density based Clustering:

Implementation:

1. Import the library necessary to perform operations, access data and plot the visualizations.
2. The entire dataset was loaded and imported.
3. The data was preprocessed and converted to a list of lists and the first column containing the row number and the second column containing the ground truth was removed. Later the data was converted to matrix form for easy accessibility.

4. Initialize lists for storing the final cluster ID and variable to check if the row was already visited for each row in data.
5. Computed and stored the distance between each point or row in the dataset to a matrix by using the distance_matrix() function from Scipy and using Euclidean distance as a metric for computing the distance. (p= 2 denotes using Euclidean Distance for distance computation).
6. The first function call is made to the dbscan() function which
7. This takes each row in the dataset and checks if it is already visited. If it is not visited then mark it now as visited and find the neighbor points of the particular point.
    a. If the number of neighbor points is less than the minimum points initialized then assign the cluster id value as zero.
    b. .Else increment the cluster id value and assign this value to the final cluster number list corresponding to the point. Also try expanding the cluster by finding the neighbor point, reachable point for each of the previous neighbor points.
    c. The loop terminates when all the points or the rows of the dataset become already visited.

8. Compute Jaccard coefficient and Rand index by using its corresponding formula and display them.

9. Now the cluster list for all the points are obtained. Check if the number of attributes in the dataset is greater than 2.

10. If "yes" then the dataset was passed through PCA dimensionality reduction to get two dimensional data. Now the clustering was visualized using the predicted labels and reduced attributes using the scatter plot from seaborn library.

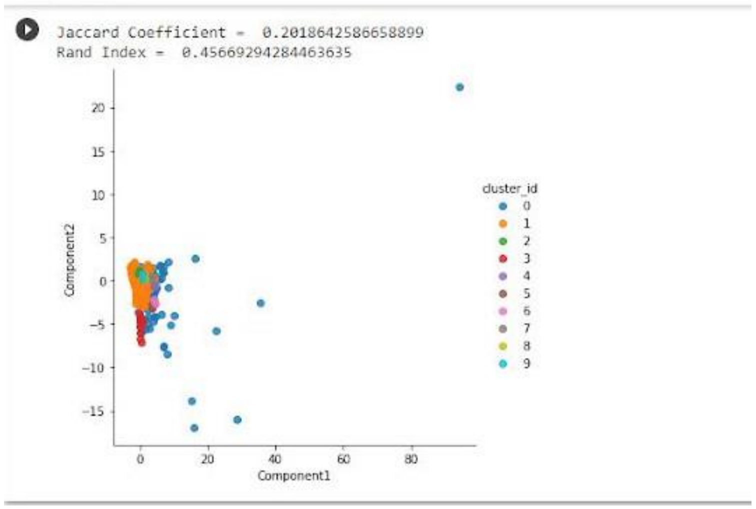
Pros: 1) Can handle clusters of different shapes and sizes
2) Resistant to noise

Cons: 1) Cannot handle varying densities
2) Sensitive to parameters

Packages used: Pandas, numpy, matplotlib, distance_matrix, PCA

Result:

iyer.txt

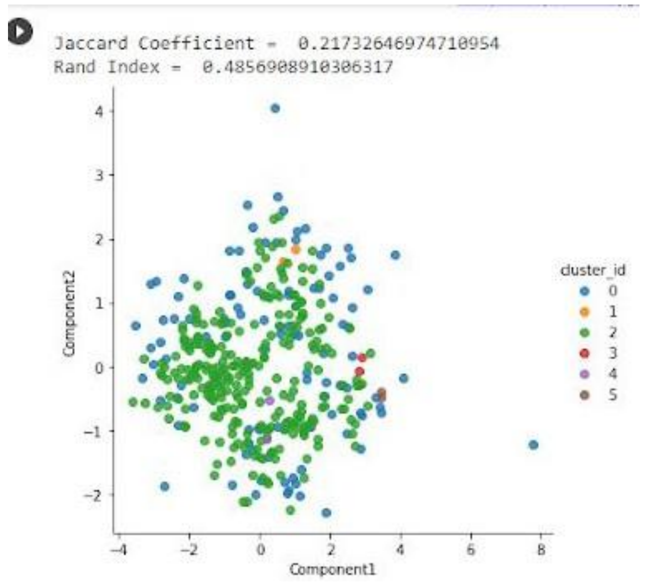


```
Jaccard Coefficient =  0.2018642586658899
Rand Index =  0.45669294284463635
```

Minimum_points = 2
Epsilon = 1.45

Jaccard coefficient:0.2018642586658899
Rand Index :0.45669294284463635

`Cho.txt`



```
Jaccard Coefficient =  0.21732646974710954
Rand Index =  0.4856908910306317
```

```
epsilon = 1.2
minimum_points = 2
```

```
Jaccard Coefficient =  0.21732646974710954
Rand Index =  0.4856908910306317
```

4. Mixture model

Mixture model Clustering:

Implementation:

1) Initialize the Phi, Sigma and Mu for the Guassian distribution functions. The convergence threshold and smoothing value are also specified.

3) Two steps are performed in each iteration: The Estep and Mstep.

 a) EStep: Compute posterior probability for each point belonging to each of the K clusters

 b) Mstep: Updates the parameters Mu, Sigma and Phi based on the probability.

3. Compute the difference between previous parameter's values and newly updated parameter's value.

4. The iteration is terminated If the difference is less than the convergence threshold or the log likelihood reaches local minimum.

5. Finally choose the cluster that gives maximum posterior probability for each point in data.

8. Compute Jaccard coefficient and Rand index by using its corresponding formula and display them.

9. Now the cluster list for all the points are obtained. Check if the number of attributes in the dataset is greater than 2.

10. If "yes" then the dataset was passed through PCA dimensionality reduction to get two dimensional data. Now the clustering was visualized using the predicted labels and reduced attributes using the scatter plot from seaborn library.

Packages used: Pandas, numpy, matplotlib, multivariate_normal

Iyer.txt

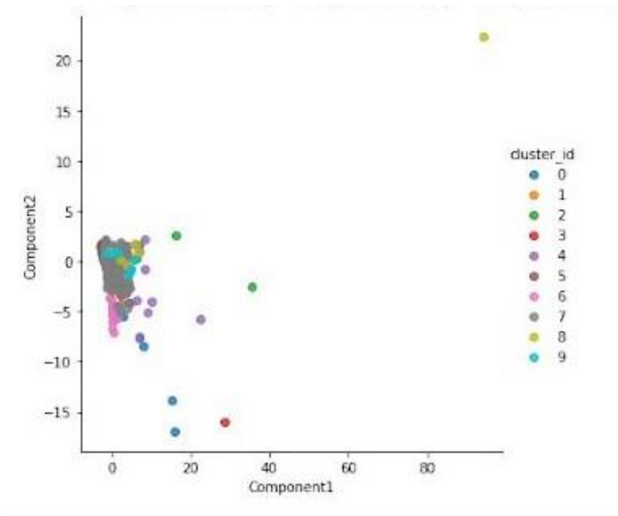
Pros: 1) Gives probabilistic cluster assignments

 2) Can handle clusters with varying shapes and sizes

Cons: 1) Overfitting issues

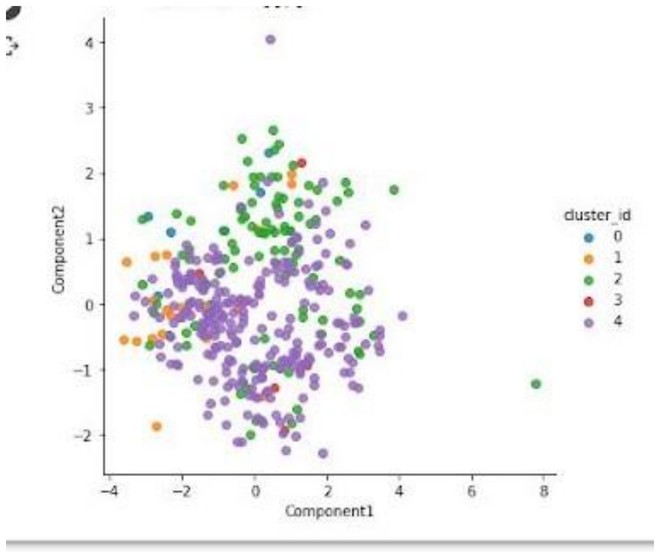
 2) Choose appropriate distributions

 3) Initialization matters



Jaccard Coefficient: 0.3163932803656748
Rand Index: 0.7157608431323399

```
Cho.txt
```



```
Jaccard Coefficient:  0.19796999709360819
Rand Index:  0.5184568713254047
```

5. Spectral Clustering

Spectral Clustering: Spectral clustering is an unsupervised machine learning clustering algorithm that groups data by finding a pattern without prior knowledge of labels as in supervised algorithms. Spectral clustering relies on graphs and proximity between data points to cluster them. There are three steps to perform spectral clustering and they are as follows:

1. Preprocessing - This step involves computing Laplacian matrix
2. Decomposition - This step involves finding eigenvalues and eigenvectors of the Laplacian matrix L and building embedded space from the eigenvectors corresponding to the d smallest eigenvalues.
3. Clustering - This step involves usage of k means clustering to produce k number of clusters using the output from the previous steps.

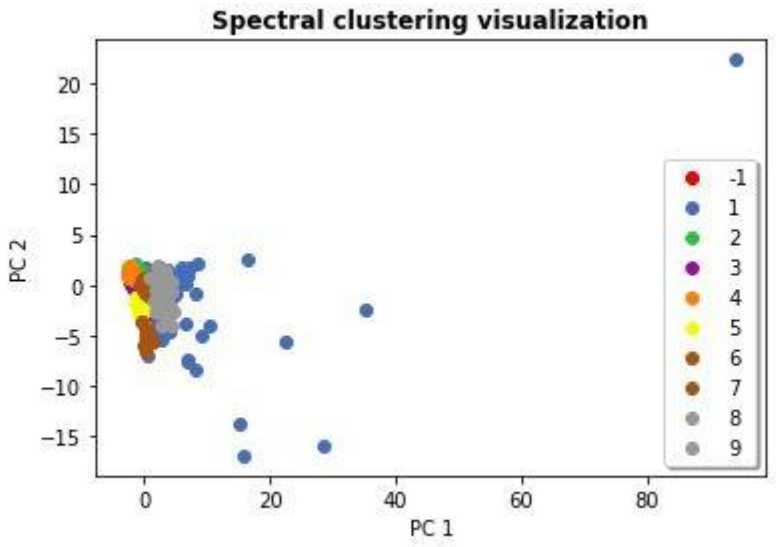
Implementation:

1. From the given data , a similarity matrix is constructed using gaussian kernel similarity as mentioned.
2. From the derived similarity matrix, a degree sum matrix is calculated
3. Using similarity matrix and degree sum matrix, we then calculate Laplacian matrix by using Laplacian matrix = Degree sum matrix - Similarity matrix
4. We then calculate eigen value and eigen vectors for that laplacian matrix.
5. K means clustering algorithm is then used to cluster points after having reduced the number of points.

Packages used: Pandas, numpy, matplotlib, sklearn for kmeans, PCA

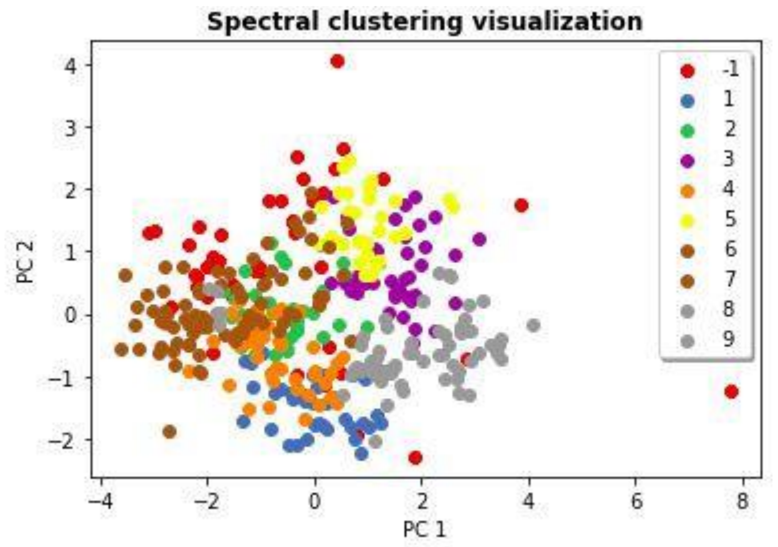
Result :

Spectral Clustering Visualization for iyer.txt



```
Jaccard Coefficient:  0.275
Rand Index:  0.823
```

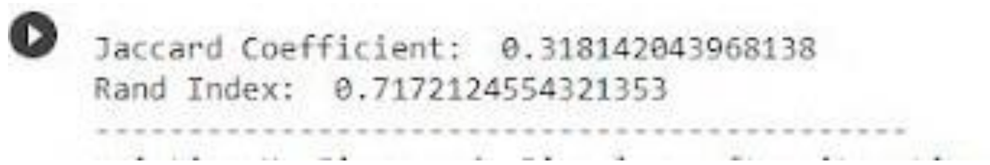
Spectral Clustering Visualization for cho.txt



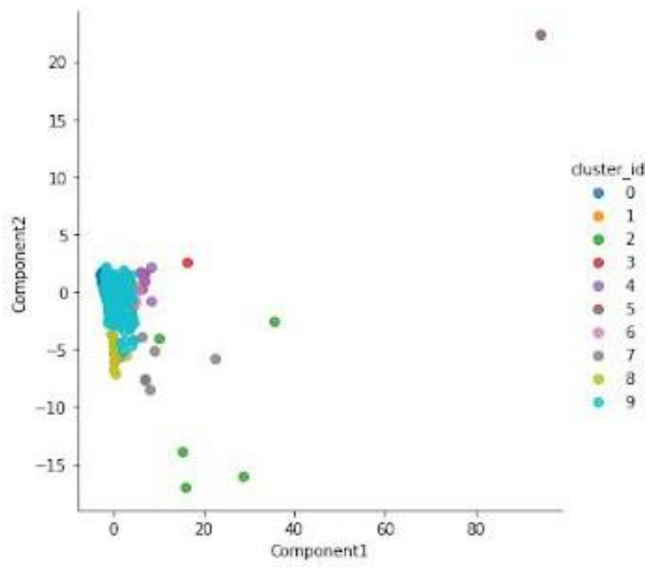
```
Jaccard Coefficient:  0.166
Rand Index:  0.763
```

Jaccard coefficient: Jaccard coefficient value is found to be 0.166 with a sigma value of 1.5 for cho.txt and 0.275 for iyer.txt
Rand Index : Rand Index value is found to be 0.763 for cho.txt and 0.823 for iyer.txt

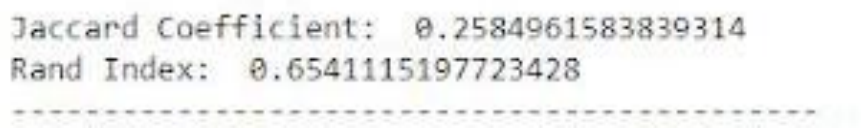
Iyer.txt C = 10

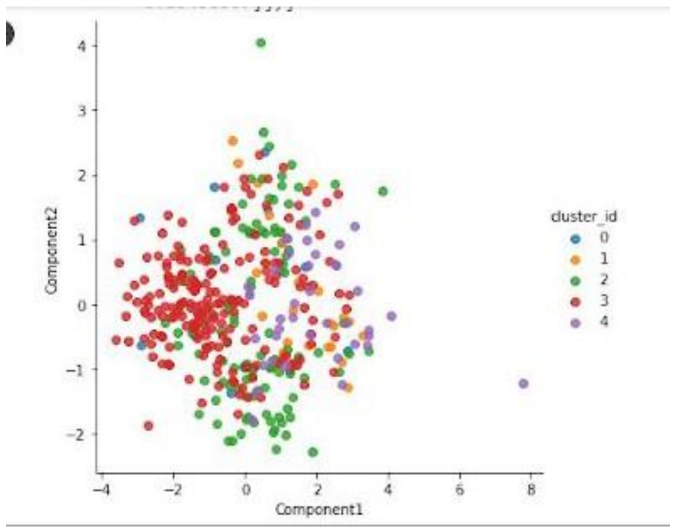


```
Jaccard Coefficient:  0.318142043968138
Rand Index:  0.7172124554321353
----------------------------------------
```



Cho.txt C=5


```
Jaccard Coefficient:  0.2584961583839314
Rand Index:  0.6541115197723428
-----------------------------------------
```


Component2
Component1
cluster_id