CSE 601:DATA MINING & BIOINFORMATICS

Clustering Algorithms

Ashutosh Ahmad Alexander	50248859	alexanda@bufalo.edu
Dilip Reddy Gaddam	50248867	dilipred@buffalo.edu
Pradeep Singh Bisht	50247429	pbisht2@buffalo.edu

K-Means clustering Algorithm:

K-Means Algorithm is used find k clusters in a given dataset. The number k is user defined. Initially we choose K random points in the given dataset and K-Means algorithm runs in two phases.

Phase 1: Calculate the distance from a point to all the K centroids and assign the point to the cluster to which the distance is minimum.

Phase 2: Recompute the new centroid of every cluster by calculating the mean of all the points in the given cluster.

Repeat these two phases until the convergence.

Algorithm Implementation:

1.Prepare the given dataset by removing un-necessary columns.

2. Initialize the value of K and select K random points in the dataset as initial centroids.

3.We initialize an another set of centroids to store the previous centroids and these are initially initialized to zero.

4.Error is calculated as distance between previous centroids and present centroids using Euclidean algorithm.

5. While error>0 we carry out the following steps:

- Calculate distance from each point to k- centroids and assign the point to the cluster with minimum distance.
- Make a copy of old centroids.
- Calculate the new centroids of each cluster by calculating the mean of all points in the cluster.
- Calculate the error using old centroids and new centroids.

Visualizations:

In order to visualize the given dataset which has large dimensions we used in built PCA algorithm of sklearn library of python.

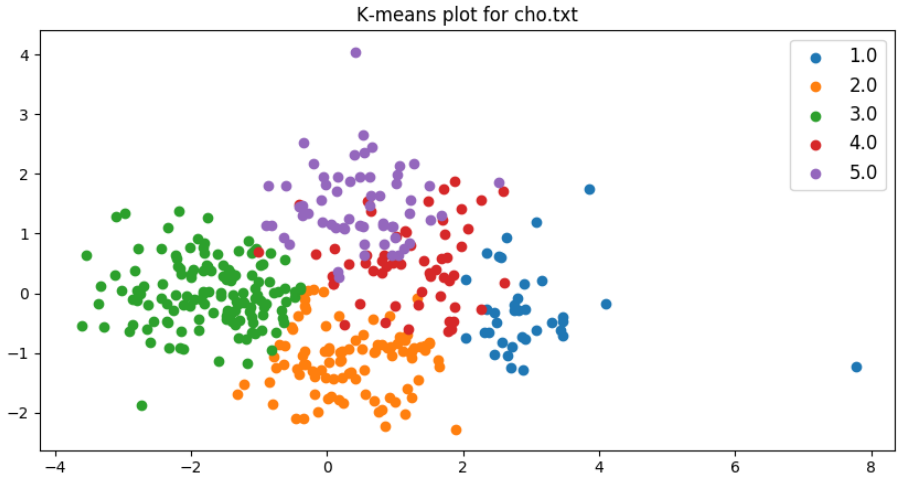

Fig:1 K-Means Plot for cho.txt with K=5

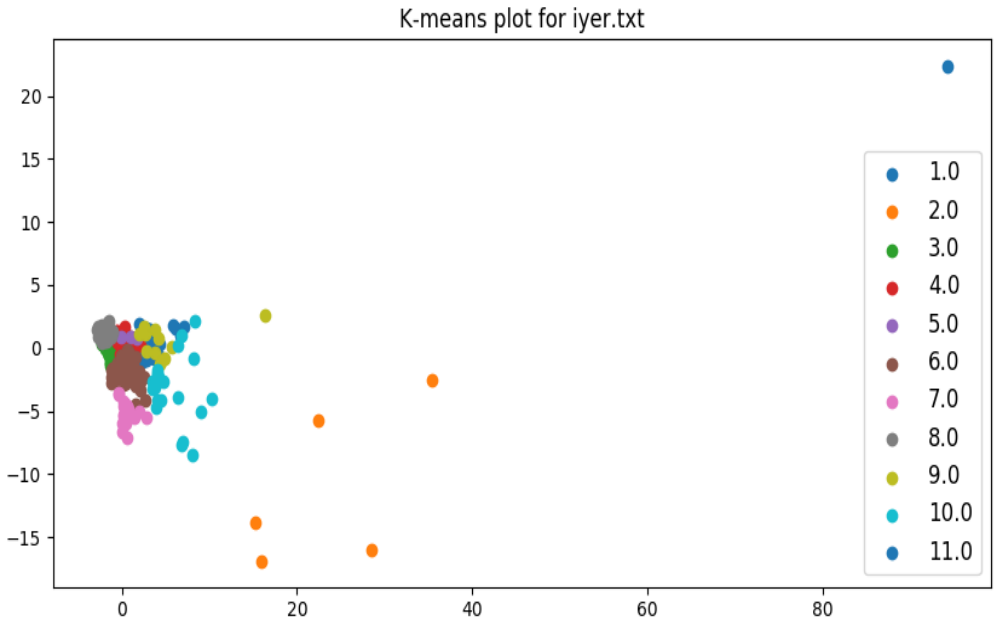

Fig 2: K-Means Plot for iyer.txt with K=11.

Validating the Results:

We use Jaccard co-efficient in order to validate the results of K-Means Algorithm. We calculate the jaccard coefficient using the ground truth of clusters and labels predicted using k-means algorithm.

$$Jaccard\ coefficient = \frac{|M_{11}|}{|M_{11}| + |M_{10}| + |M_{01}|}$$

Jaccard Coefficient for **cho.txt:** 0.4064

Jaccard Coefficient for **iyer.txt:** 0.2917

PROS of K-Means:

1.K-Means is easy to implement.

2.K-means is an efficient clustering algorithm with the complexity of O(tkn) where n=number of objects, t=number of iterations and k = number of clusters. Generally t,k are very small compared to n.

3.K-Means produce tighter clusters when compared to any other clustering algorithms.

CONS of K-Means:

1.Difficult to predict the number of clusters.

2.Initial centroids affect the algorithm.

3.works poorly when clusters are not similarly sized or clusters of varying density.

4. There is a possibility for empty clusters.

Agglomerative Hierarchical Clustering:

1.Agglomerative Clustering is a bottom-up approach where each node is considered as a single individual cluster.

2.Compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters.

3.In single linkage hierarchical clustering, the distance between two clusters is defined as the shortest distance between two points in each cluster and it is used as similarity measure.

Algorithm Implementation:

1.Prepare the given dataset by removing un-necessary columns.

2. Initialize the value of K. Where K determines the number of clusters to be formed and also tells when the algorithm to be terminated.

3.Calcuate a N*N distance matrix where N represents numbers of items in the dataset and each index i,j in matrix represents distance between ith record and jth record.

4.Select two records with the minimum distance and merge those two records. It reduces the matrix size by 1.

5.The distance matrix is recomputed for remaining points using minimum distance between two of the merged points. For example, if points 2 and 5 are merged in this step. Distance of all other points will be calculated w.r.t to cluster (2,5) using, d((2,5), P) = min(d(2,P),d(5,P)

6.Repeat steps 4,5 until the size of distance matrix equals K.

Since only two clusters are merged in one iteration the time complexity for this algorithm is O(n2) for all the points to be merged into one cluster.

Visualizations:

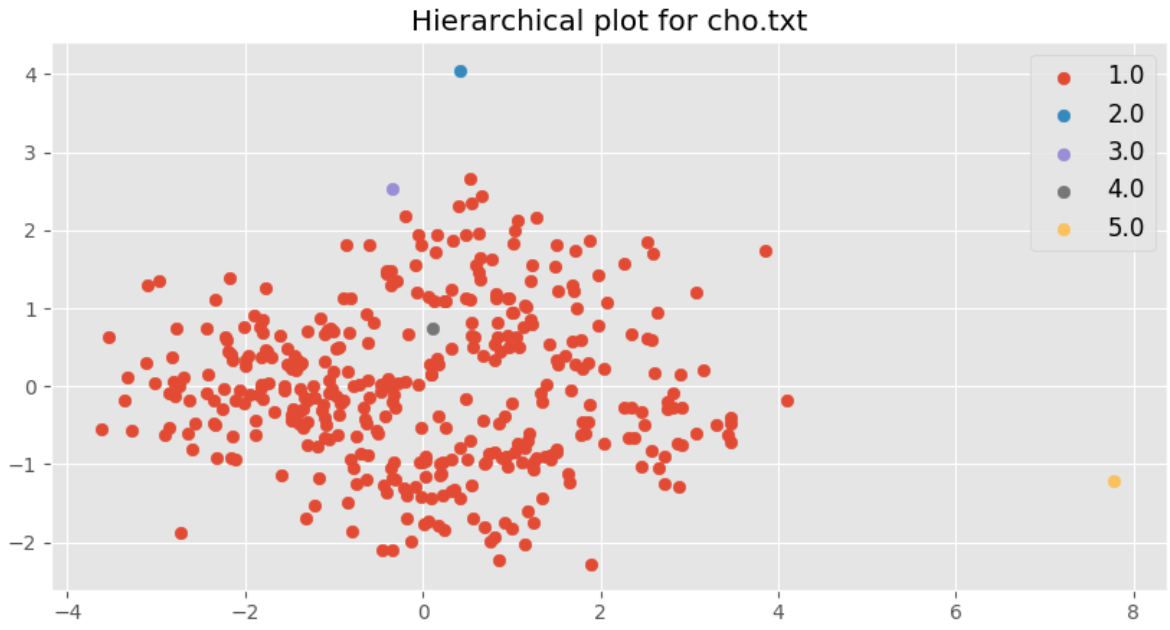

Fig 3:Hierarchical Plot for cho.txt with number of Clusters=5

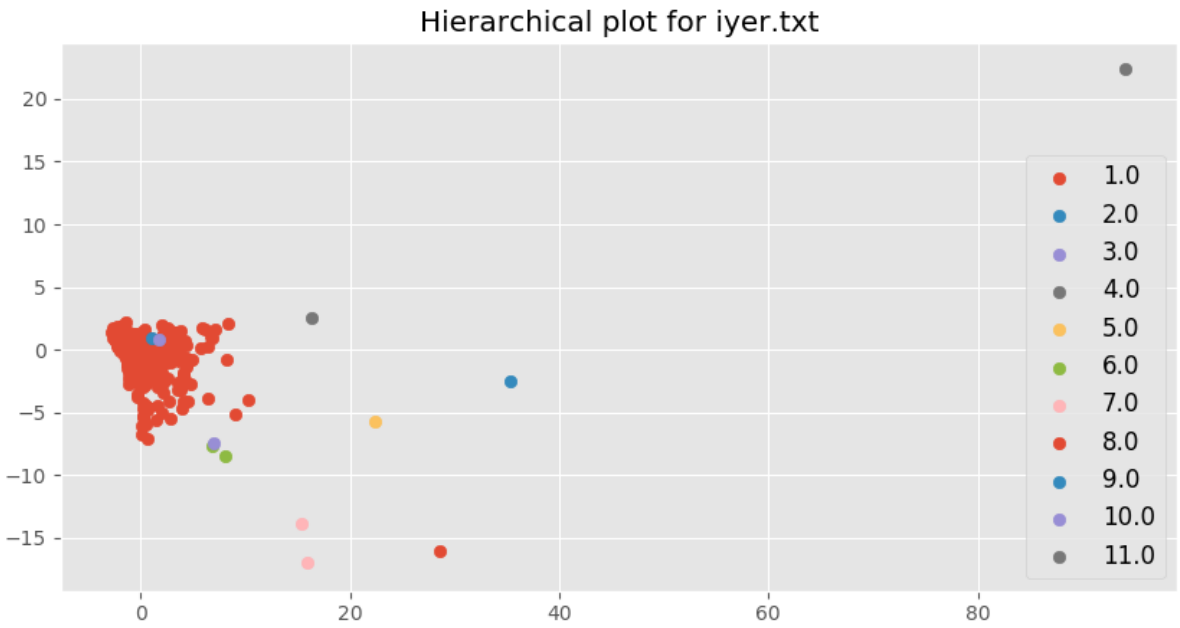

Fig 4: Hierarchical Plot for iyer.txt with number of Clusters=11

Validating the Results:

Jaccard co-efficient for cho.txt: 0.228.

Jaccard co-efficient for iyer.txt: 0.158.

PROS of Hierarchical Clustering:

1.Any number of clusters can be produced by cutting the dendrogram at particular level.

2.Hierarchical clustering corresponds to meaningful taxonomies.

3.It is shown to capture concentric clusters.

4. It provides hierarchical relations between clusters.

CONS of Hierarchical Clustering:

1. computationally expensive and sensitive to outliers.

2.Decision once made cannot be undone.

3. No objective function is directly minimized.

4. Difficulty handling different sized clusters and irregular shapes.

DENSITY-BASED SPATIAL CLUSTERING AND APPLICATION WITH NOISE (DBSCAN):

DBSCAN is a density based algorithm. It locates regions of high density that are separated from other regions of low density, where density = number of points within a specified radius.

Points in dataset can be classified into three types:

Core points: Points having more than minimum number of points in its neighborhood. These points are present at the interior of a cluster. Any core points that are close enough – within a distance of given radius – are present in the same cluster.

Border points: Points having less than minimum number of points in its neighborhood, but still present in neighborhood of other core points. Any border point that is close enough to a core point is present in the same cluster as the core point.

Noise: Any point that is not a core or border point and also having less than minimum number of points in its neighborhood. Noise are not present in any cluster.

• **Density-reachable**: An object q is directly density-reachable from object p if q is within the εneighborhood of p and p is a core object.

• **Density-connectivity**: An object p is density-connected to object q with respect to ε and minimum number of points if there is an object o such that both p and q are density-reachable from o with respect to ε and minimum number of points.

DBSCAN Algorithm and Implementation:

DBSCAN algorithm has 2 parameters

Ɛ-Neighborhood: object within Ɛ from an object.

Minpts: minimum number of points to be present in the Ɛ-Neighborhood.

Steps:

1.Prepare the given dataset by removing un-necessary columns.

2.Initialize the Ɛ-Neighborhood and minpts which are passed as parameters to algorithm.

3.The initial clusters of every point is assigned to 0.

4.For every point in dataset if the point is not visited then calculate the number of points using ‘regionQuery’ in the Neighborhood and if count of points is less than minpts then mark the point has Noise.

5.If the count of points in the neighborhood is greater than or equal to minpts then classify the label of that point to the clusterId by incrementing the clustered and these point is passed to ‘expandcluser’ function which marks all the neighborhood points of this cluster with this clusterId and marks them visited.

6.Repeat this process till all the points are visited and labelled with cluster id’s.

Visualizations:

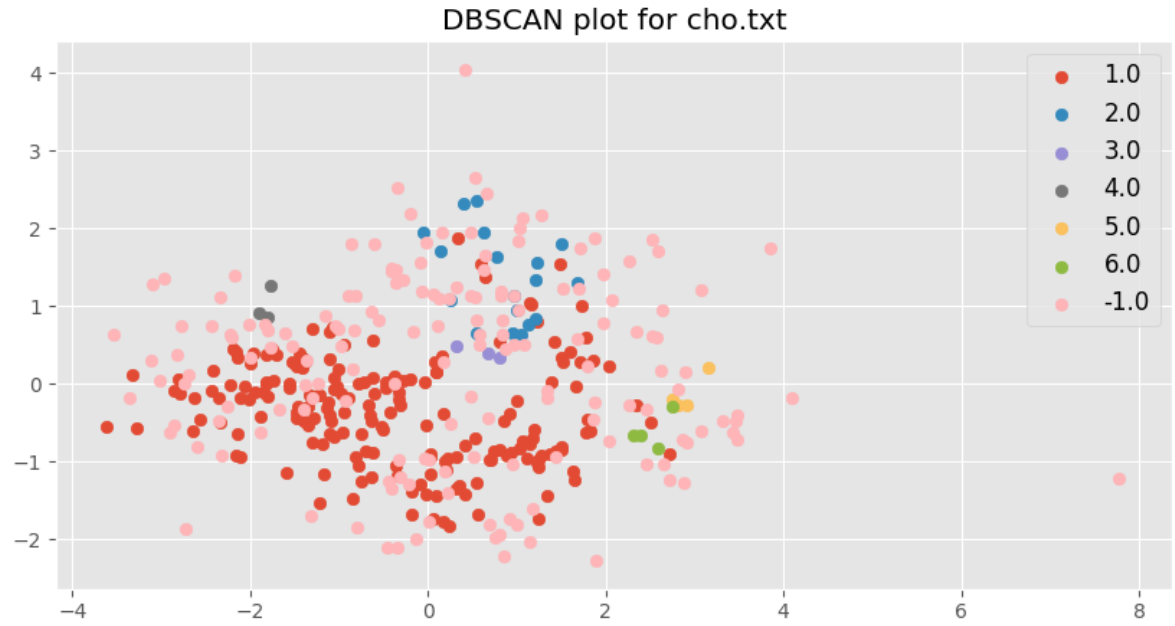

Fig 5: DBSCAN Plot for cho.txt with Ɛ=1 and minpts=3.

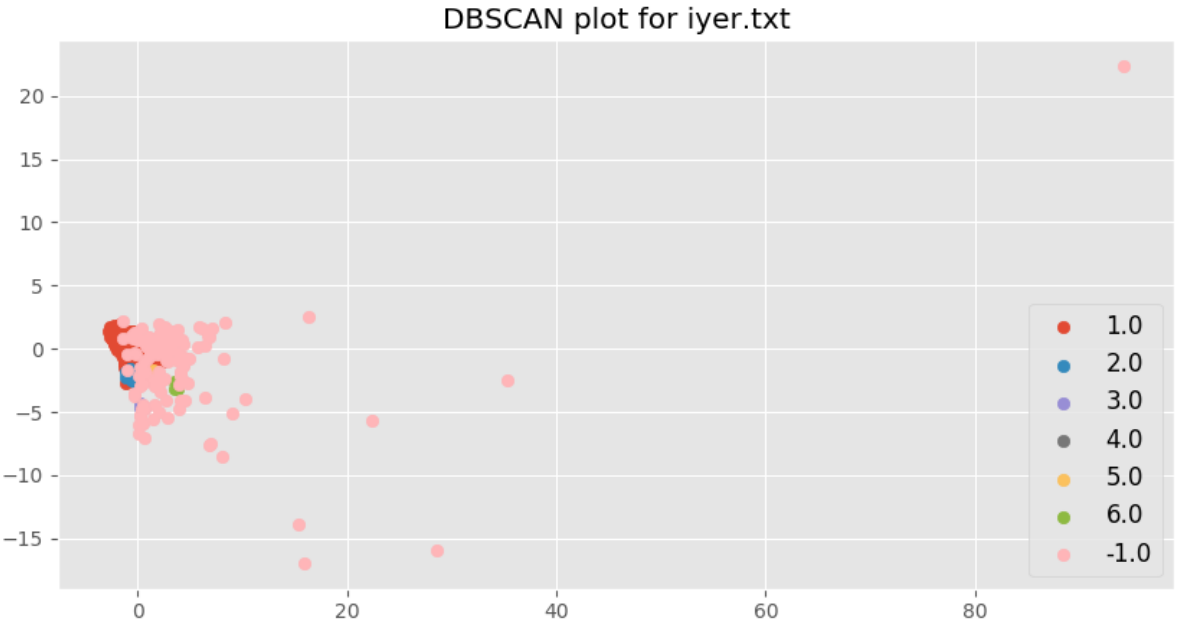

Fig 6: DBSCAN Plot for iyer.txt with Ɛ=1 and minpts=3

Validating the Results:

Jaccard co-efficient for cho.txt: 0.2044.

Jaccard co-efficient for iyer.txt: 0.2835.

PROS of DBSCAN:

1.It has a time complexity of O(n2) and space complexity of O(n), where n is the number of data points.

2.It doesn't require to mention number of clusters.

3. DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster.

4.This algorithm is resistant to noise.

5.DBSCAN can handle clusters of different shapes and sizes.

CONS of DBSCAN:

1.It is not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed.

2.The quality depends on the distance measure used in the function regionQuery, used to identify the neighbours.

3.If the data and scale are not well understood, choosing a meaningful distance threshold ε can be difficult.

K Means Hadoop:

Hadoop MapReduce is a software framework for writing programs to process large datasets in-parallel manner.

A MapReduce job splits the input data set in to chunks of datasets which are processed by mapper task in parallel processing. The result of the mapper task is than sorted and serve as the input to reducer task and give the desired result.

Algorithm Implementation:

1.Prepare the given dataset by removing un-necessary columns.
2. Initialize the value of K and select K random points in the dataset as initial centroids.
3. write the initial centroids to a file "cluster.txt" and the preprocessed data set to the "input.txt".
4. Map – In the mapper task we are reading the cluster.txt file to get the centroids values and generate the tab separated key-value pairs where key is cluster centroid index and value is the pair of dataset index and its features.
5. The output is then sorted and send to the reducer.
6. Reduce - the reducer read line by line and across similar dataset index sum the features till the new cluster index is coming then prints the tab separated previous cluster index along with dataset index and feature values.
7. Output of all the reducers are then used to generate the cluster.txt and input.txt for the next iteration.
8. Step 3 to 7 are repeated till we reached convergence when the old and new centroids are equal.

Output:

Total run: 15

Jaccard similarity for cho.txt: 0.4064

Total run: 46

Jaccard similarity for iyer.txt: 0.2917

Visualization:

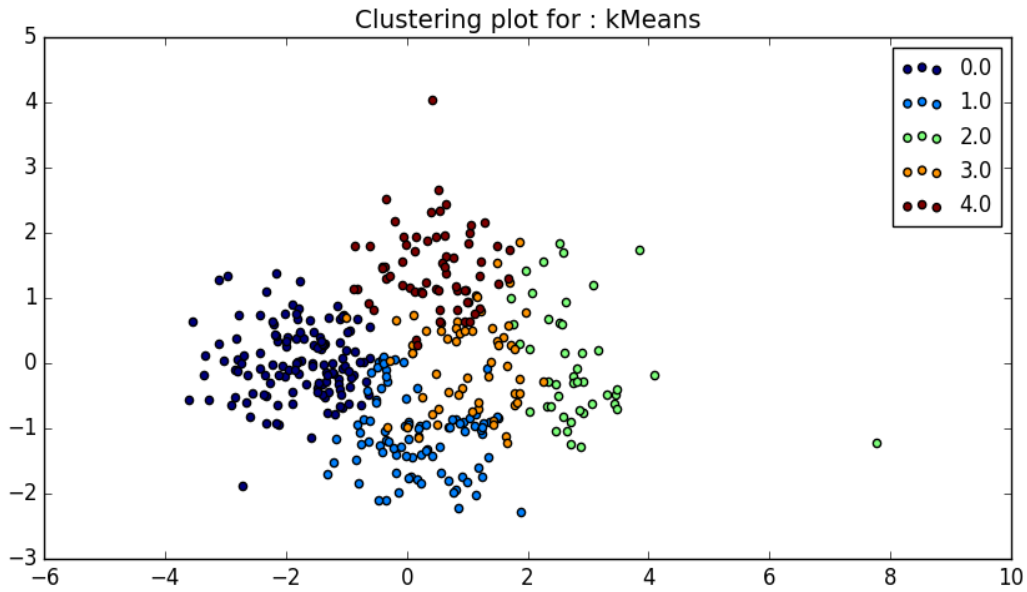

Fig 7: Means Plot for cho.txt with K=5

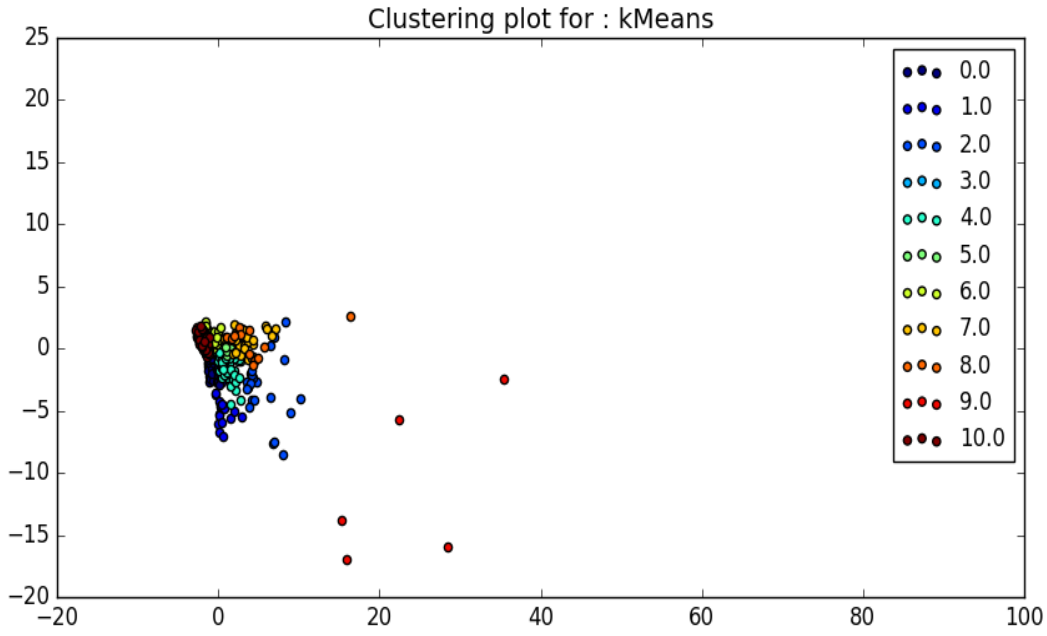

Fig 8: K-Means Plot for iyer.txt with K=11

Analysis of K Means on Map Reduce

In the map reduce implementation of the K means the time taken is more than the k means because it is running on the single node cluster. To improve the performance we can do the following :

1) we can place all the cluster.txt file to the hdfs system so that access to the mappers is fast.
2) For larger dataset will be running on the multiple node than the performance will be improved.
3) To increase the mapper and reducer based on the number of k clusters than the computation can be parallelized and increased significantly.

Jaccard Coefficient:

Clustering Algorithm	**Cho.txt**	**Iyer.txt**
K means	0.4064	0.2917
Hierarchical	0.228	0.158
DBSCAN	0.2044	0.2835
K means Hadoop	0.4064	0.2917

The result for the K means algorithm on the normal and Hadoop distribution are same for similar initial centroid as expected.