Assignment 4

Jigar Chandra (Chandra.67) Gaurav Shah (Shah.889)

Goal of the Assignment

To demonstrate various clustering techniques on the Reuters document dataset based on the feature vectors generated in Assignment 1.

Distance Metrics:

We attempted running our algorithms with two different distance metrics:

1. **Manhattan distance**
   The Manhattan distance between two items is the sum of the differences of their corresponding components.

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

2. **Euclidean distance**
   Euclidean distance is the actual point-to-point Cartesian distance between two points in a vector space. For an n-dimensional space, Euclidean distance (d) between points p and q is calculated as:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

These metrics were used to compute distance between a pair of articles based on the feature vector generated.

K-means clustering

This is a convergence based algorithm that works in two steps per iteration. The algorithm expects the number of clusters K as an input. We start with K random mean points, one mean for each cluster. The means are represented as a vector of the same dimensionality as the feature vector of the articles. Each article acts as a point for this algorithm.

In each iteration, the first step is to find the closest cluster mean for each point and assign the point to that cluster. Each cluster gets a set of points nearby its mean. In the second step, all cluster means are recomputed. This is straightforward by taking a mean of each dimension and forming a mean vector.

The clustering algorithm converges when in two successive iterations no points or only a small number of points are reassigned to different clusters. This can also be represented in terms of change in the cluster means.

We have used the Orange library for implementing this algorithm.

Results and Performance:

Entropy:

For finding out how the algorithm performed, we used entropy as the performance metric.

Computing entropy:

Entropy is first individually computed for each cluster and then a weighted average is taken based on the number of points in the clusters. The entropy for the cluster is computed as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

for n class values and b=2.

We found that as the number of clusters increase, entropy gradually decreases, meaning a better quality of separation of articles.

For K-means clustering algorithm, we computed the following entropies:

# Clusters	Entropy for Euclidean Distance	Entropy for Manhattan Distance
2	3.44398295708	3.26022447805
8	3.16275468649	3.16168079334
16	2.86684278239	2.87032127003
32	2.82207970737	2.82207279299
64	2.70244179526	2.70109704559
80	2.63560732218	2.64177427703
100	2.57085885736	2.56781025785

According to the following plot, it can be observed that Manhattan distance produces slightly lower entropy and hence is a slightly better distance metric for clustering given data with respect to entropy.

Scalability:

The running time is roughly directly proportional to the number of clusters K provided as input. We took results for K=2, 4, 8, 16, 32, 64. It is found that running time is directly proportional to the number of clusters.

# Clusters	Time for Euclidean Distance (seconds)	Time for Manhattan Distance (seconds)
2	9.56123534782	3.92417416682
8	11.8539132114	9.88658780019
16	32.1417110191	25.9337313511
32	86.8370767621	74.3929538916
64	109.658534173	75.7519171981
80	185.652399699	121.644757497
100	253.815954578	150.539725502

Skew:

Skewness, that is deviation in size of clusters, decreases as the value of k increases. Thus, skewness is observed most, when k = 2 and gradually increases upto k = 100. We used standard deviation with respect to the cluster size, as a measure for skewness. A standard deviation value of 883 was observed for (Manhattan)k=2, while for (Manhattan)k=64 , it decreased to 245.

Hierarchical Clustering

The clustering is implemented in python using the orange library. Orange library requires the distance matrix and the computation of the distance matrix(11367 X 11367) takes a very long time to compute since we haven't employed any parallelism mechanism.

The algorithm takes approximately 30 to 35 minutes to run completely

Various steps in computation

Step1 - Distance matrix : As stated earlier this is the most time consuming step and is done using a standard orange library command. The distance matrix is computed from the .tab file that we have generated and is used further for clustering

Step 2 - Clustering: Heirarchial clustering has been done based on MIN LINK (a.k.a.SINGLE). Also Orange returns the clustering in the form of a tree so we have accessed the cluster not according to the number of clusters but till what depth in the tree would we like to consider the clusters . So if we mention the depth as say 10, then the algorithm will go down the tree from the root node upto 10 levels and will consider all the clusters upto that level. Also we have shown the number of clusters available at the depth specified.

Please note: This depth from root node will be referred to as level in the remaining documentation.

Step 3 - Entropy Calculation:

Entropy is first individually computed for each cluster and then a weighted average is taken based on the number of points in the clusters. Now, each topic's/place's weight is summed over all the records in that cluster. Once we get a vector of total weights of all class labels, we normalize it to find probabilities of each topic/place.

Level	# Clusters	Entropy for Euclidean Distance	Entropy for Manhattan Distance
10	65	3.7713046951	3.77289276407
20	230	3.64948619885	3.65404244029
30	495	3.62035322273	3.64424058075
40	860	3.52724731619	3.54868471897
50	1325	3.49868944697	3.50872896856

Please note: As entropy was decreasing by a very small amount, so we have increased the level by 10 each time so that we can record a visible decrease in entropy.

Scalability:

Since step 3 works directly on distances, the choice of metric affects its running time. Computation time for Manhattan distances is less than that of Euclidean since it uses magnitudes and does not involve computations based on taking the squares and the square root, like Euclidean. Below is an estimate of the running time:

Stage	Time in seconds
1 – Euclidean	1503.92192015
2 – Manhattan	1331.68288888

Please note: this time is the combined time taken to build the distance matrix and perform clustering.

Skew:

Skewness, that is deviation in size of clusters, decreases as the level (depth from root node) increases. Thus, skewness is observed most, when level = 5 and gradually decreases upto level = 50. We used standard deviation with respect to the cluster size, as a measure for skewness. A standard deviation value of 282 was observed for level=5, while for level=10 , it decreased to 234.

Individual Contributions

Jigar implemented the K-mean clustering algorithm while Gaurav implemented Hierarchical clustering.