Name : Divya Khiani

SJSU ID : 015273245

Report on Bisecting K-Means

- PseudoCode for Bisecting k-means

```
Step 1 : Converting the 'train.dat' file to compressed sparse row matrix of shape 8580 * 126355 with term frequencies in each document as the value.
Step 2 :  Transform the csr matrix using term frequency times inverse document frequency.
Step 3 : Dimensionality reduction is applied on the transformed matrix to achieve a dimensionality of 8580*500
Step 4 : Bisecting k-means is applied on this reduced data. In bisect_k function the data is recursively divided until required number of clusters are achieved
Step 5 : Apply k-means on the data points to get 2 clusters.
Step 6 : In k-means algorithm, findClusters function finds the similarity between data points and the initialized centroid, it returns the index of the cluster in which it belong.
Step 7 : After this findCentroids function reinitializes the centroid.
Step 6 : Compare the error sum of squares of the two clusters formed and divide the cluster with higher SSE and add the other one in list of clusters.
Step 7 : Repeat from step 6 until desired number of clusters are achieved.
Step 8 : Bisect_k function returns list of clustered points and error sum of squares for all the clusters.
Step 9 : Plot the average error sum of squares for k=3 to k=21 and observe the trend.
```

- Observation

  After plotting the average error sum of squares for k = 3 to k = 21 we observe the following trend. We observe that when k = 3 the SSE is highest and as we increase the number of clusters it gets on decreasing. But considering a very high number of clusters will result in overfitting the data. Using the elbow method we can use k=7 for the data.

  Choosing the right k is the most important thing in clustering algorithms. Choosing a very small k may result in underfitting of data while choosing a higher k value will result in overfitting of data. To keep a balance between these two we choose k using elbow method.

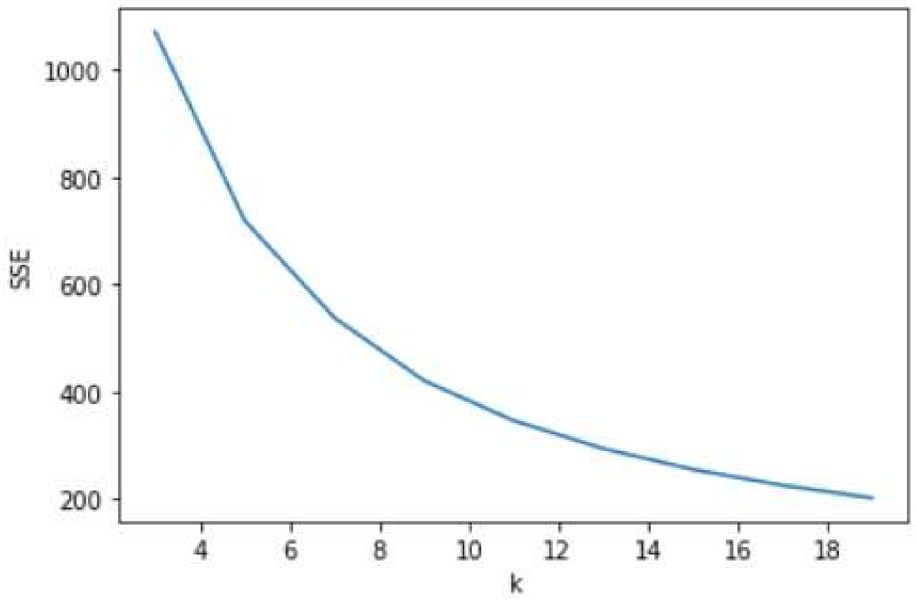

Fig : Average of distance from centroid vs Number of clusters

- Measures Used for proximity calculation and Dimensionality reduction

  TFIDF transformation is used to supress the impact of most frequently occurring words. tf-idf(t, d) = tf(t, d) * idf(t) and idf(t) = log [n / df(t)] + 1. L2 norm was used to calculate the tfidf so that cosine similarity is calculated between two data points.

  For dimensionality reduction, **TruncatedSVD** (arpack algorithm) is used. Using this we were able to achieve to reduce the dimensionality from 8580*126355 to 8580*500. Truncated SVD works best with tf-idf matrices hence it is been used here. The dataset gives more efficient result when ARPACK algorithm is used. While using randomized algorithm the accuracy was around 63% but with ARPACK it increased a bit to 65%.
  Why TruncatedSVD over PCA ?
  PCA would operate on entire matrix for the output while in Truncated SVD we can specify the number of columns and also it is efficient in handling sparse matrices.

  For calculation of error sum of squares **Euclidean distance** is used, which also defines the proximity function of the cluster. If the cluster is tightly bound the SSE will be lower, while if the cluster is widely spread out the SSE will be higher.