**SHARAD DIXIT**

**CMSC 676 INFORMATION RETRIEVAL**

1. **OBJECTIVE**

The objective of this assignment was to build a similarity matrix and further use the similarity matrix in order to perform agglomerative clustering using average link method. With each step of the algorithm, whichever documents are merged (clustered) together are shown and further the clustering ceases when no two clusters (or documents) have similarity greater than 0.4.

**2.) ARCHITECTURE**

The task of this project first involved development of similarity matrix which was developed by following steps:

* Term frequency and Inverse document frequency calculated for each token in each document and then on the basis of that tf\*idf was calculated.
* After calculation of tf\*idf, weight normalization was done which involved tf\*idf weight of a token by the square root and summation of square of each token weight in the respective document.
* After normalization of weights, cosine similarity was calculated for each pair document where each possible pair was considered and both normalized weights were multiplied with each other.
* The cosine similarity calculated was stored in matrix (known as similarity matrix) where both row and column representing the documents and each element representing the pairs cosine similarity.

After development of similarity matrix, the matrix was used to perform agglomerative clustering using average link grouping method. The following steps were done to perform agglomerative clustering.

* The maximum element was found in the similarity matrix was found first by traversing through the matrix which basically was the document pair with the highest similarity score. This pair of the documents formed the first cluster.
* After creation of the first cluster, the matrix is reconstructed with the first cluster considered as a single element and through group average link method the average of the similarity score is done with each of the element in order to find the cosine similarity with all the other documents and the cluster.
* After reconstruction of the new similarity matrix with the first cluster as an element, then again, the process of agglomerative clustering is repeated, and the maximum value is found by traversing through the matrix and furthermore new cluster are formed or the documents are added to the cluster already present according to the cosine score in the similarity matrix.
* The above process is repeated until there is no cosine similarity value of 0.4 greater than available in the similarity matrix.

1. **METHODOLOGY**

The project methodology is divided in two following steps:

**# Building of Similarity Matrix**

* For Building of the similarity matrix, it is done through the continuation of the Project 2 where we developed the tf\*idf weight for each term.
* The output of the Project 2 is therefore then used to calculate the cosine similarity between all the documents and then the cosine similarity is saved to the matrix with rows and columns both as document which the cosine similarity belongs to.
* After development of the similarity matrix, the matrix is serialized as a File.ser through Java serialization to the local disk so that we don’t have to calculate the similarity matrix again and again and we can directly use the matrix in order to do agglomerative clustering.

**# Agglomerative Clustering**

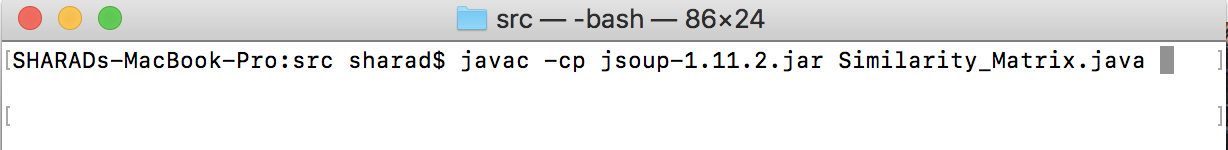
* For performing the agglomerative clustering, we need to deserialized the similarity matrix file and then use that file to perform the clustering.
* Therefore the File.ser is deserialized and stored in an 2D array.
* The retrieved similarity matrix is then used to find the highest similarity matrix which is done by traversing through the matrix and then the documents having highest similarity forms the first cluster.
* Each cluster is stored in
* After creation of the first cluster, the matrix is reconstructed with the first cluster considered as a single element and through group average link method the average of the similarity score is done with each of the element in order to find the cosine similarity with all the other documents and the cluster.
* After reconstruction of the new similarity matrix with the first cluster as an element, then again, the process of agglomerative clustering is repeated, and the maximum value is found by traversing through the matrix and furthermore new cluster are formed or the documents are added to the cluster already present according to the cosine score in the similarity matrix.
* Therefore, the process is repeated until the threshold of 0.4 is reached and clustering ceases.
* Since each cluster is stored in an ArrayList therefore the index of ArrayList serve as label of the clusters

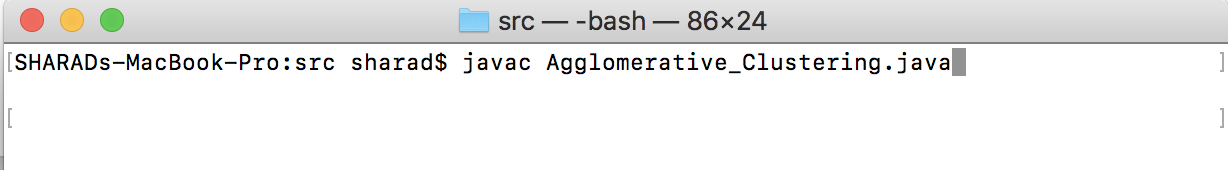
1. **EVALUATION OF PROGRAM**

* The efficiency of the program is very optimized as the similarity matrix is serialized and not calculated again and again and hence save us the task of calculating the tf\*idf again and again which is a tedious task and hence the performance of the program is very optimized.
* The agglomerative clustering is also very apt as the developed similarity matrix is correct and also since the matrix is present in the ArrayList therefore is traversed very quickly.

1. **OUTPUT**
2. **Compilation of Java source code:**

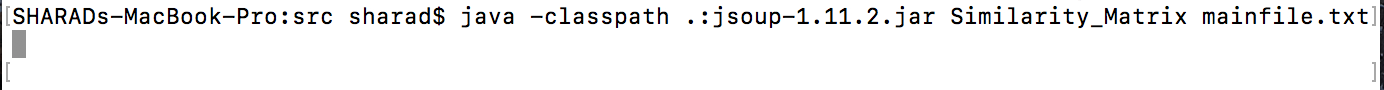
* Compilation of source code of similarity matrix.

****

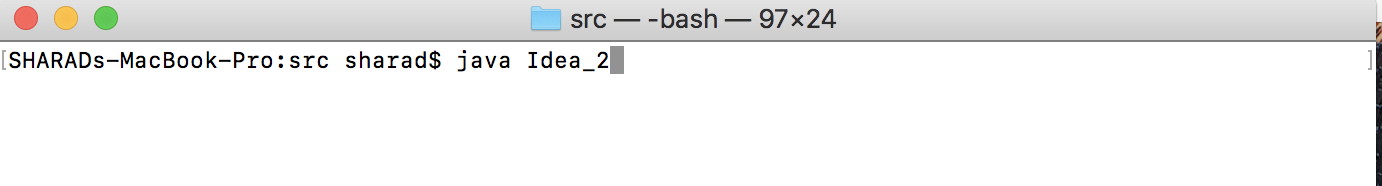
* Compilation of source code of Agglomerative clustering****

1. **Running of Java Class file**

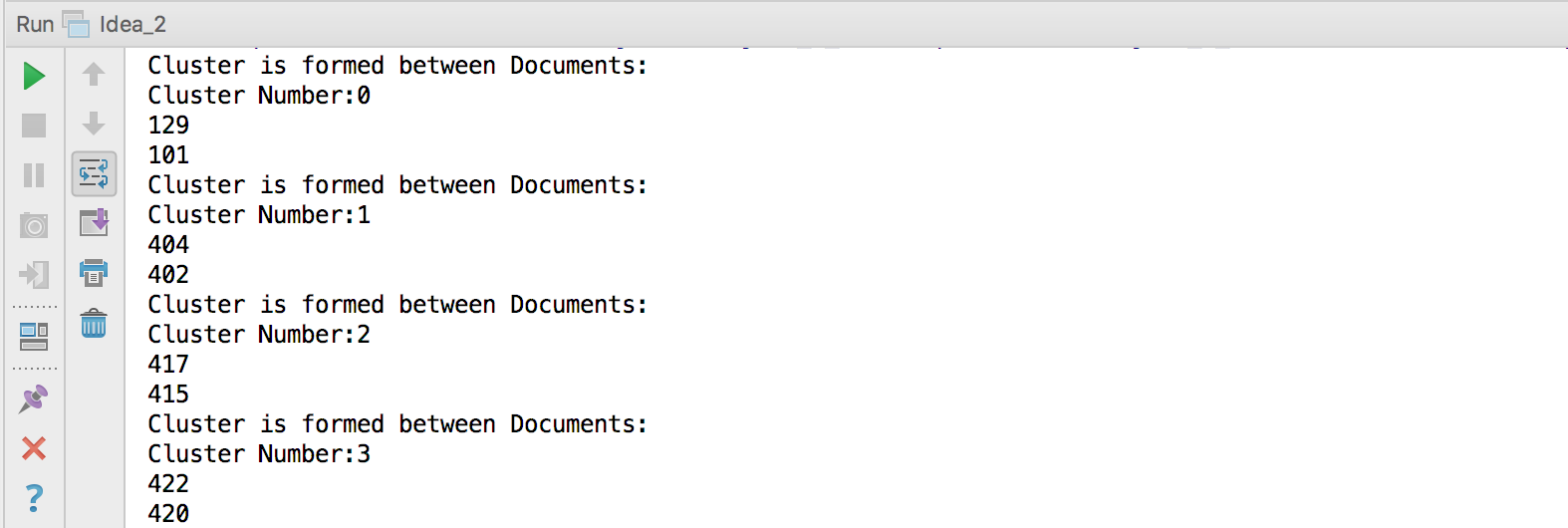
* Running of similarity matrix class file

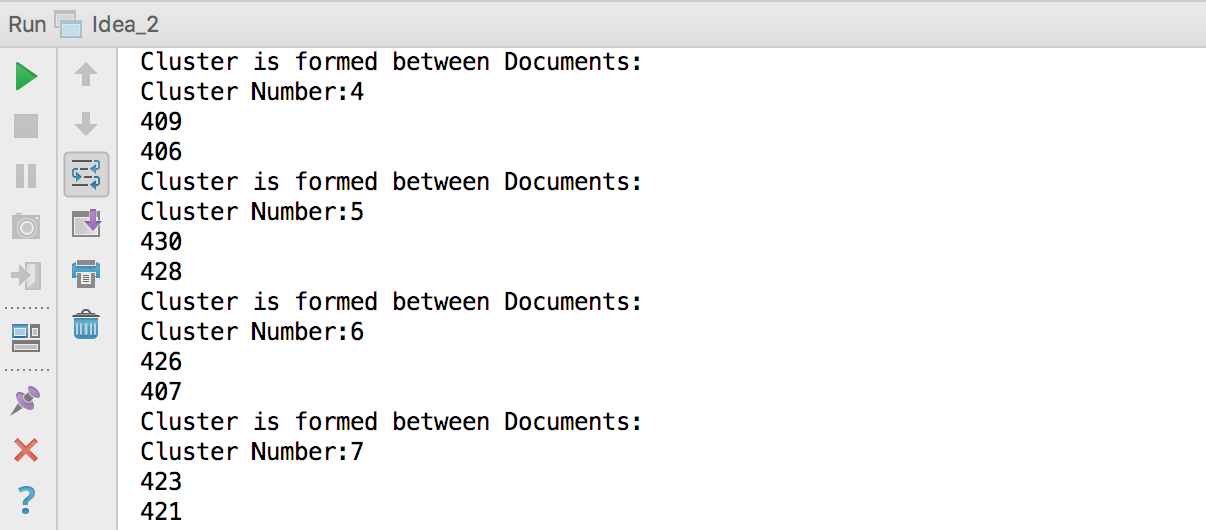
****

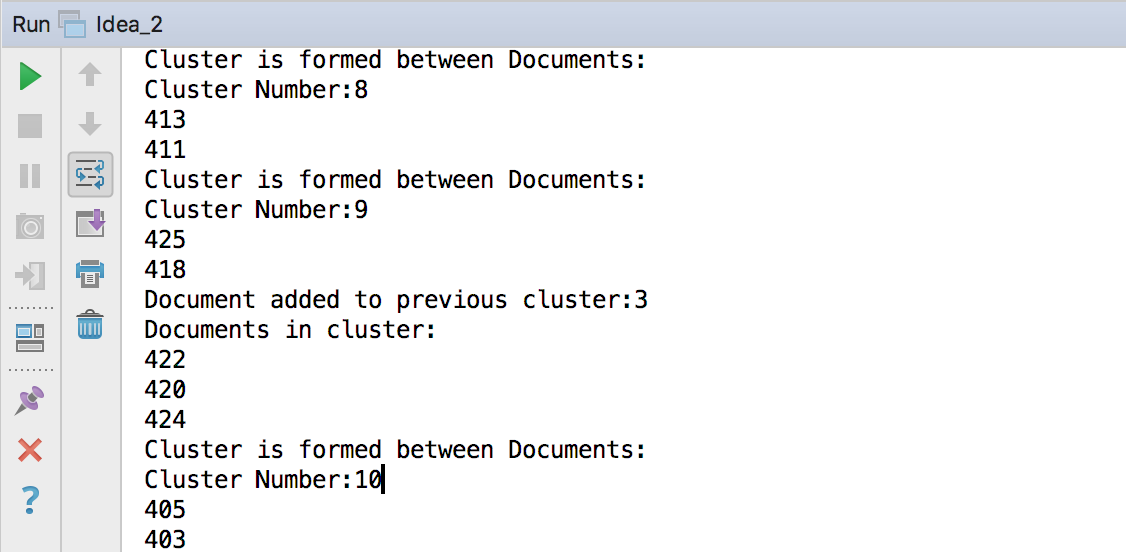
* Running of Agglomerative class file

****

1. **Output**

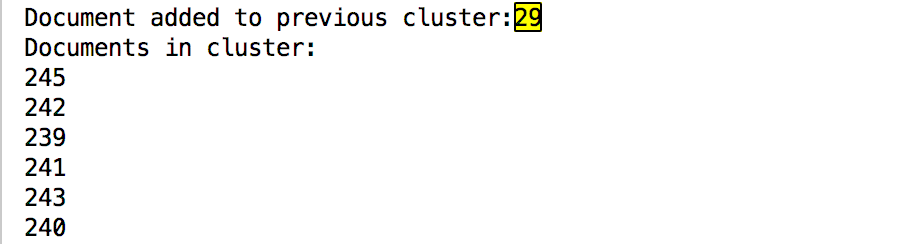
****

****

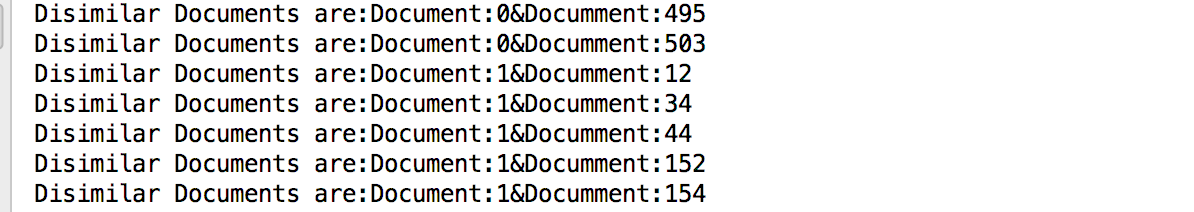
****

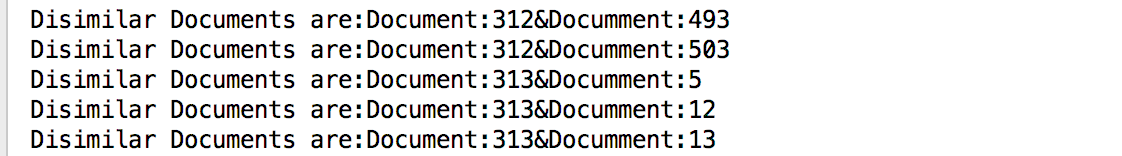
The above is the output of the agglomerative clustering.

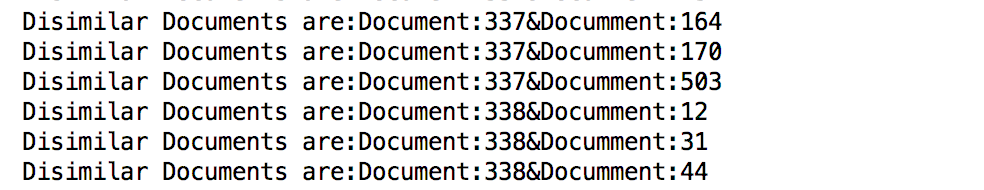
1. The first cluster formed is between Documents 129 and Documents 101 and hence these two documents are most similar according my similarity matrix developed and according to my cosine similarity of these two documents.
2. The label which the cluster is assigned is the cluster number, this is obtained as since all the cluster are getting stored in an ArrayList therefore the index of the ArrayList serve as a Cluster label in our case.
3. Cluster number 29 has the maximum number of the documents when the clustering ceases and hence therefore the centroid of cluster 29 servers as the centroid of the whole corpus, where first document 245 serves as cluster first added and centroid.



1. Since a lot of documents in the whole corpus were not similar to any other documents as the entries of the similarity matrix for those documents were zero. Therefore, below is list of some of those documents.





Therefore, above are some example of documents that are most dissimilar that is they don’t have any token in common.

1. **Limitation with Reason**

The limitation to my approach is that the whole matrix has to be traversed when we need to find the maximum element present in the matrix, which could be very costly when the matrix is very big and can result in a lot of performance degradation. This could be improved by applying any other search algorithm for traversing other than linear search for the maximum element in the matrix.

1. **Working of Program**
2. **Pre-Requisite**

# Input the mainfile.txt as the input for my project, that contains the filepath for the all the input files is needed in order develop the similarity matrix, and after that then use that matrix to perform agglomerative clustering.

1. **Developing Environment**

# Mac OS with 16GB RAM

1. **Submission Package**

# Report

# File.ser (Java Serialized file for the similarity matrix)

The above file can be directly use for entering in the “Idea\_2.java” file for to check the clustering.

# Output file for the agglomerative clustering (AggCluster.txt)

# Similarity\_Matrix.java (source code for the similarity matrix)

# Idea\_2.java (source code for the Agglomerative Clustering)