**NAME** : Patwari Venkatesh Nandkumar

**ROLL:** 407B001

**ASSIGNMENT NO. 2**

**CODE:**

import java.io.\*;

import java.util.\*;

public class Assignment2 {

public static void main(String[] args) throws IOException {

BufferedReader stdinpt = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the number of Tokens:");

int noOfTokens = Integer.parseInt(stdinpt.readLine());

System.out.println("Enter the number of Documents:");

int noOfDocuments = Integer.parseInt(stdinpt.readLine());

System.out.println("Enter the threshold:");

float threshold = Float.parseFloat(stdinpt.readLine());

System.out.println("Enter the Document Token Matrix:");

int[][] input = new int[noOfDocuments][noOfTokens];

for (int i = 0; i < noOfDocuments; i++) {

for (int j = 0; j < noOfTokens; j++) {

System.out.println("Enter (" + i + ", " + j + "):");

input[i][j] = Integer.parseInt(stdinpt.readLine());

}

}

SinglePassAlgorithm(noOfDocuments, noOfTokens, threshold, input);

}

private static void SinglePassAlgorithm(int noOfDocuments, int noOfTokens, float threshold, int[][] input) {

int[][] cluster = new int[noOfDocuments][noOfDocuments + 1];

ArrayList<Float[]> clusterRepresentative = new ArrayList<>();

// Initialize first document as the first cluster

cluster[0][0] = 1;

cluster[0][1] = 0;

int noOfClusters = 1;

Float[] temp = convertIntArrToFloatArr(input[0]);

clusterRepresentative.add(temp);

for (int i = 1; i < noOfDocuments; i++) {

float max = -1;

int clusterId = -1;

// Find the most similar cluster

for (int j = 0; j < noOfClusters; j++) {

float similarity = calculateSimilarity(convertIntArrToFloatArr(input[i]), clusterRepresentative.get(j));

if (similarity > threshold && similarity > max) {

max = similarity;

clusterId = j;

}

}

// Assign the document to a cluster

if (max == -1) { // Create a new cluster

cluster[noOfClusters][0] = 1;

cluster[noOfClusters][1] = i;

noOfClusters++;

clusterRepresentative.add(convertIntArrToFloatArr(input[i]));

} else { // Add to an existing cluster

cluster[clusterId][0]++;

int index = cluster[clusterId][0];

cluster[clusterId][index] = i;

clusterRepresentative.set(clusterId, calculateClusterRepresentative(cluster[clusterId], input, noOfTokens));

}

}

// Output clusters

for (int i = 0; i < noOfClusters; i++) {

System.out.print("\nCluster " + i + ": ");

for (int j = 1; j <= cluster[i][0]; j++) {

System.out.print(cluster[i][j] + " ");

}

}

}

private static Float[] convertIntArrToFloatArr(int[] input) {

int size = input.length;

Float[] result = new Float[size];

for (int i = 0; i < size; i++) {

result[i] = (float) input[i];

}

return result;

}

private static float calculateSimilarity(Float[] a, Float[] b) {

float similarity = 0;

for (int i = 0; i < a.length; i++) {

similarity += a[i] \* b[i];

}

return similarity;

}

private static Float[] calculateClusterRepresentative(int[] cluster, int[][] input, int noOfTokens) {

Float[] representative = new Float[noOfTokens];

Arrays.fill(representative, 0.0f);

for (int i = 1; i <= cluster[0]; i++) {

for (int j = 0; j < noOfTokens; j++) {

representative[j] += input[cluster[i]][j];

}

}

for (int i = 0; i < noOfTokens; i++) {

representative[i] /= cluster[0];

}

return representative;

}

}

**OUTPUT:**

base) venkatesh@Venkateshs-MacBook-Air Java % javac Conflation.java

(base) venkatesh@Venkateshs-MacBook-Air Java % java Conflation

base) venkatesh@Venkateshs-MacBook-Air Java % javac Assignment2.java

(base) venkatesh@Venkateshs-MacBook-Air Java % java Assignment2

Enter the number of Tokens:

3

Enter the number of Documents:

3

Enter the threshold:

10

Enter the Document Token Matrix:

Enter (0, 0):

1

Enter (0, 1):

0

Enter (0, 2):

1

Enter (1, 0):

2

Enter (1, 1):

1

Enter (1, 2):

0

Enter (2, 0):

2

Enter (2, 1):

3

Enter (2, 2):

1

Cluster 0: 0

Cluster 1: 1

Cluster 2: 2 %

(base) venkatesh@Venkateshs-MacBook-Air Java %