CS5425 Assignment 2 Task 2 Report by A0212221E – Vincent Andrian Chandra

Note: Euclidean distance is changed to return the square root of the sum between 2 points instead of just the sum.

Task 2 contains 4 main steps:

1. Pre-processing
2. K-means algorithm
3. Results

1. Pre-processing

To incorporate the data into k-means algorithm, first a pre-processing is done to convert the data into vectors. Pre-processing step consists of joining questions and answers data based on post ID, computing highest answer score for each question, converting the questions into vectors of (D*X, S) where D is 50,000, X is the index of the question tag, and S is the highest score. These steps are represented by the functions / methods from the code template, respectively ***groupedPostings,*** ***scoredPostings,*** and ***vectorPostings***.

After the vectors are obtained, ***sampleVectors*** function is called to generate 45 initial centroids. The centroids generation is implemented using Kmeans++ method where 1 initial centroid is chosen randomly, then compute each vector's distance to its nearest centroid and search for a vector that has the highest distance to its nearest centroid. That particular vector will be designated as another centroid, and repeat the process until 45 centroids are assigned.

2. K-means Algorithm

K-means algorithm is executed to get the cluster results with convergence criteria of sum of each vector / data point distance to its nearest centroid < 20 and a maximum of 120 k-means iteration.

First thing to be done in the algorithm is checking whether the clusters have converged by calculating the sum of all distances from the vectors' centroid. If they have converged, then the k-means function will return a key-value pair where the key is the final centroid and the value is an iterable of vectors belonging to the key / centroid. On the other hand, vectors are grouped to their closest centroid using the provided ***findClosest*** function which returns closest centroid index and an update to the centroids position according to the average of the grouped vectors is done if none of the convergence criteria has been reached. Finally, the algorithm will move on to the next iteration.

3. Result

This result is achieved by having the parameters of ***DomainSpread*** of 50,000, ***kmeansKernels*** of 45, ***kmeansEta*** of 20, and ***kmeansMaxIterations*** of 120.

```
Centroid no: 1, Position: (0,1635), Size: 2, Average: 1635, Median: 1635
Centroid no: 2, Position: (550000,6), Size: 1338, Average: 6, Median: 4
Centroid no: 3, Position: (50000,1498), Size: 1, Average: 1498, Median: 1498
Centroid no: 4, Position: (500000,4), Size: 1673, Average: 4, Median: 3
Centroid no: 5, Position: (700000,2), Size: 397, Average: 2, Median: 1
Centroid no: 6, Position: (500000,93), Size: 25, Average: 93, Median: 73
Centroid no: 7, Position: (0,83), Size: 355, Average: 83, Median: 71
Centroid no: 8, Position: (350000,2), Size: 14508, Average: 2, Median: 1
Centroid no: 9, Position: (150000,104), Size: 177, Average: 104, Median: 87
Centroid no: 10, Position: (50000,5904), Size: 1, Average: 5904, Median: 5904
Centroid no: 11, Position: (50000,701), Size: 6, Average: 701, Median: 720
Centroid no: 12, Position: (200000,2), Size: 46658, Average: 2, Median: 1
Centroid no: 13, Position: (250000,82), Size: 167, Average: 82, Median: 63
Centroid no: 14, Position: (250000,609), Size: 7, Average: 609, Median: 583
Centroid no: 15, Position: (650000,6), Size: 521, Average: 6, Median: 3
Centroid no: 16, Position: (150000,434), Size: 23, Average: 434, Median: 396
Centroid no: 17, Position: (400000,794), Size: 4, Average: 794, Median: 777
Centroid no: 18, Position: (450000,289), Size: 1, Average: 289, Median: 289
Centroid no: 19, Position: (0,320), Size: 51, Average: 320, Median: 302
Centroid no: 20, Position: (250000,3), Size: 23240, Average: 3, Median: 2
Centroid no: 21, Position: (350000,1042), Size: 1, Average: 1042, Median: 1042
Centroid no: 22, Position: (50000,123), Size: 258, Average: 123, Median: 97
Centroid no: 23, Position: (250000,1016), Size: 2, Average: 1016, Median: 1016
Centroid no: 24, Position: (100000,1), Size: 40798, Average: 1, Median: 1
Centroid no: 25, Position: (0,775), Size: 18, Average: 775, Median: 735
Centroid no: 26, Position: (400000,3), Size: 12376, Average: 3, Median: 1
Centroid no: 27, Position: (450000,3), Size: 3036, Average: 3, Median: 2
Centroid no: 28, Position: (350000,196), Size: 54, Average: 196, Median: 170
Centroid no: 29, Position: (200000,110), Size: 269, Average: 110, Median: 89
Centroid no: 30, Position: (150000,3), Size: 22491, Average: 3, Median: 2
Centroid no: 31, Position: (0,1414), Size: 2, Average: 1414, Median: 1414
Centroid no: 32, Position: (600000,2), Size: 1815, Average: 2, Median: 1
Centroid no: 33, Position: (250000,3748), Size: 1, Average: 3748, Median: 3748
Centroid no: 34, Position: (100000,68), Size: 244, Average: 68, Median: 54
Centroid no: 35, Position: (400000,242), Size: 35, Average: 242, Median: 215
Centroid no: 36, Position: (150000,1379), Size: 1, Average: 1379, Median: 1379
Centroid no: 37, Position: (100000,434), Size: 10, Average: 434, Median: 383
Centroid no: 38, Position: (200000,524), Size: 14, Average: 524, Median: 466
Centroid no: 39, Position: (50000,2), Size: 49152, Average: 2, Median: 1
Centroid no: 40, Position: (50000,1244), Size: 1, Average: 1244, Median: 1244
Centroid no: 41, Position: (300000,3), Size: 6993, Average: 3, Median: 1
Centroid no: 42, Position: (300000,133), Size: 51, Average: 133, Median: 117
Centroid no: 43, Position: (150000,978), Size: 4, Average: 978, Median: 994
Centroid no: 44, Position: (300000,591), Size: 3, Average: 591, Median: 612
Centroid no: 45, Position: (0,2), Size: 46663, Average: 2, Median: 1
```

3.1 Insights

Using the convergence criteria of sum of vector distance to its nearest centroid is not very ideal because the convergence criteria for the sum is 20 while in reality the sum is enormous due to D*X where D is 50,000 and X only ranges to 15. Thus, the k-means will always only converge after > 120 iterations.

Also, vector distribution for the clusters is not uniform / uneven. This problem may be able to be solved by experimenting with number of initial centroids.

45 centroids were initially chosen randomly from the vectors. However, it didn't produce satisfactory result where some clusters have no vector / data point due to choosing bad centroids. Thus, k-means++ choosing method is implemented instead and as seen on the result above, at least none of the cluster is empty.

3.2 Different Parameters

With 15 ***k*** instead of 45, the vectors are much more evenly distributed. 20 ***k*** still have some clusters with size of tens or below, so 15 ***k*** is better to gain useful insights.

```
Centroid no: 1, Position: (200000,3), Size: 46941, Average: 3, Median: 1
Centroid no: 2, Position: (550000,6), Size: 1338, Average: 6, Median: 4
Centroid no: 3, Position: (700000,2), Size: 397, Average: 2, Median: 1
Centroid no: 4, Position: (450000,3), Size: 3037, Average: 3, Median: 2
Centroid no: 5, Position: (150000,4), Size: 22696, Average: 4, Median: 2
Centroid no: 6, Position: (500000,5), Size: 1698, Average: 5, Median: 3
Centroid no: 7, Position: (50000,3), Size: 49419, Average: 3, Median: 1
Centroid no: 8, Position: (600000,2), Size: 1815, Average: 2, Median: 1
Centroid no: 9, Position: (650000,6), Size: 521, Average: 6, Median: 3
Centroid no: 10, Position: (300000,4), Size: 7047, Average: 4, Median: 1
Centroid no: 11, Position: (400000,4), Size: 12415, Average: 4, Median: 1
Centroid no: 12, Position: (350000,3), Size: 14563, Average: 3, Median: 1
Centroid no: 13, Position: (100000,2), Size: 41052, Average: 2, Median: 1
Centroid no: 14, Position: (0,3), Size: 47091, Average: 3, Median: 1
Centroid no: 15, Position: (250000,4), Size: 23417, Average: 4, Median: 2
```

Decreasing ***DomainSpread*** will give the same effect as increasing the number of ***k*** where vectors are not evenly distributed in clusters. Since ***DomainSpread*** is low, the vectors are a lot more tightly coupled. Increasing the number of iterations will roughly yield the same result.

3.3 Performance Suggestion

Important thing to note is before entering the k-means function, which is a recursive function, the vectors are cached first to avoid repeating vector transformations from the beginning at each iteration. Otherwise, without cache the performance will be much slower due to lazy transformations.

As stated before, choosing 45 random initial centroids is not the best approach. Instead, it is recommended to follow Kmeans++ algorithm to generate the centroids.