**Assignment2 Task 1 Report**

Jin Shuyuan A0162475B

1. **Description of my code**

* Input data processing

Convert each line in the QA\_data to a Posting

Group the question and answers together

Choose the highest score for each question

* K-means clustering

Sample kmeansKernels vectors as the initial centroids

Cluster the vectors based on the centroids

Compute the accumulated distance in each cluster and add them as the total distance

Use the total distance to check whether the process is converged

If not, recursively call kmeans again; if yes, stop the recursion and return means

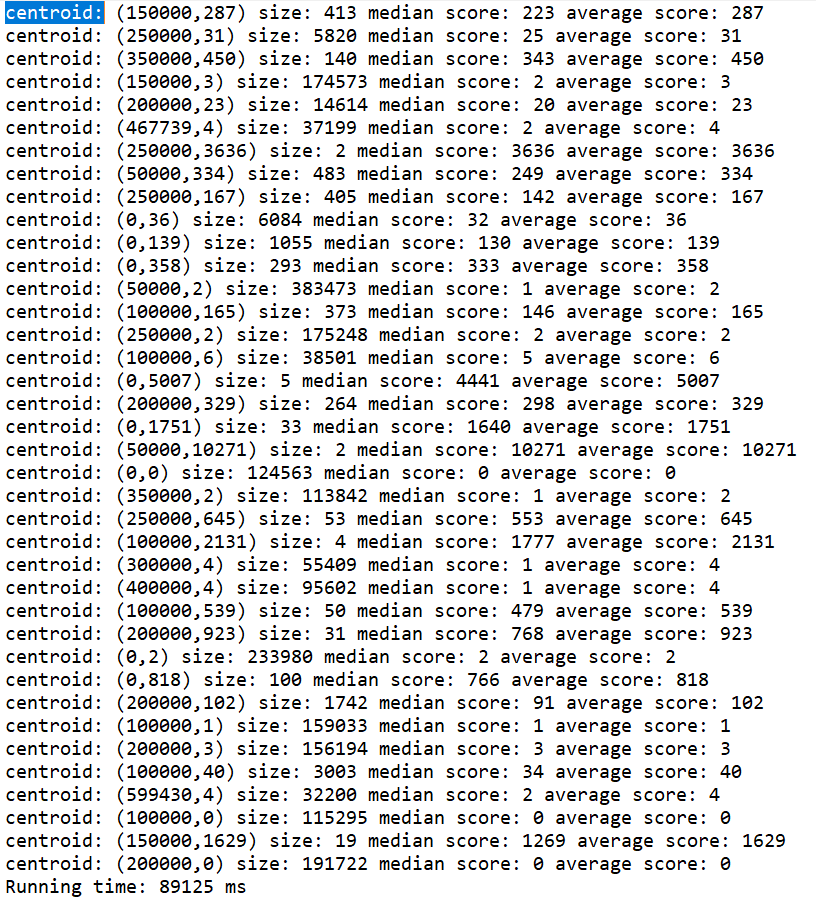
* Format the final result

Generate the cluster centroid, size of every cluster, median of score and average of score for each cluster

Print the result

1. **Analysis of the results**

* Screenshot of the final output in output.txt



* Convergence

Plot of sum of squared error (SSE) with respect to iteration of one experiment.

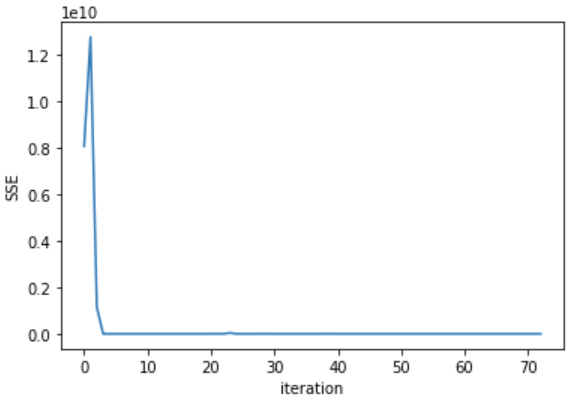


Fig 1 SSE w.r.t. iteration

From the above plot, we can see that SSE drops drastically initially and then converges at iteration 53. So the training process has good convergence ability.

* Visualization of clusters

The visualization of the final centroids:

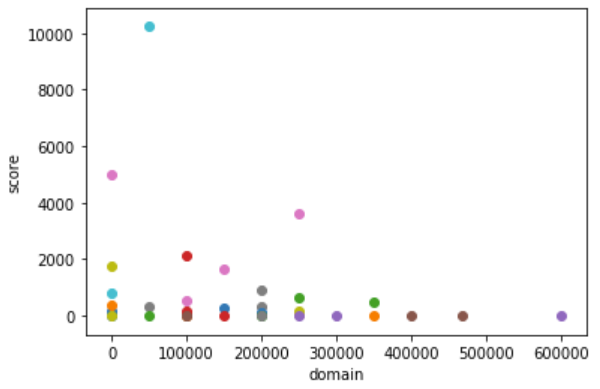


Fig 2 final centroids

To reflect the size of each cluster in the figure, I plot each centroid with its size proportional to the cluster size. We can see that most of the questions/answers belong to the first 5 topics which are "Machine-Learning", "Compute-Science", "Algorithm", "Big-Data", "Data-Analysis", "Security". This implies that these are relatively popular topics.

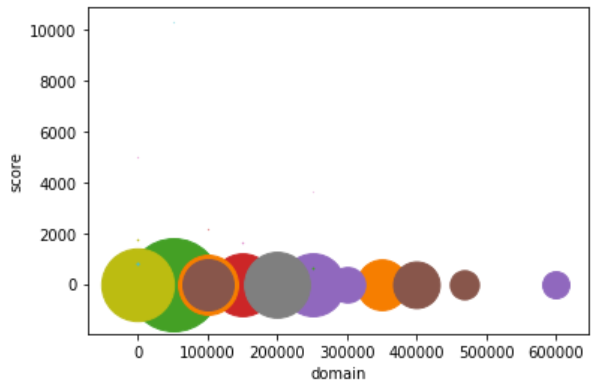


Fig 3 centroids with size reflecting cluster size

The following is a histogram of average scores of the clusters. We can see that most clusters have average scores below 1000. It is consistent with real-world case that is most answers cannot get a high score.

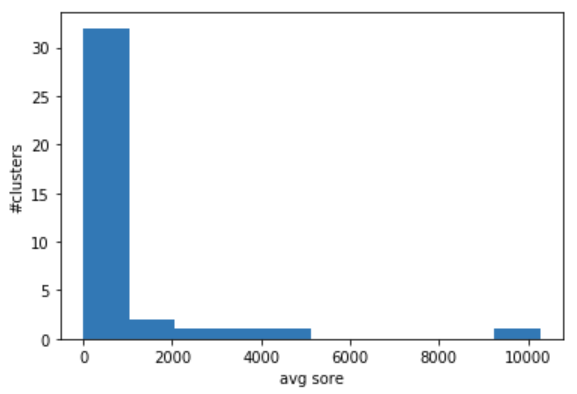
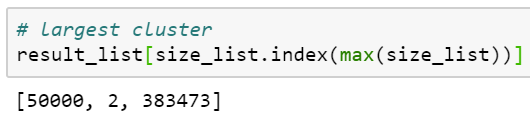


Fig 4 average score histogram

* Basic analysis of results
* The largest cluster: belongs to “Computer-Science” topic, the average score is only 2

This implies that Computer Science is a hot topic but the quality of the answers generally is not good.

 [domain, avgScore, size]

* The cluster with highest average score is “Computer-Science”, and the average score is 10271.

There are 2 question/answers fall in this cluster. I guess these 2 questions are very important questions in Computer Science and draws wide attention.



1. **Analysis of the parameters**

In this assignment task, the performance metric is SSE between old centroids and new centroids.

|  |  |
| --- | --- |
| **Parameters** | **Impact** |
| DomainSpread | Performance: larger DomainSpread increases SSE, making it harder to meet the convergence requirement.  Results: larger DomainSpread increases the distance between the clusters in the result. So the clusters are sparser. |
| kmeansKernels | Performance: SSE lowers with larger kmeansKernels. But a very large kmeansKernels leads to very dense clusters where the clusters are close to each other.  Results: larger kmeansKernels results in larger number of clusters. |
| kmeansEta | Performance: smaller kmeansEta makes the k-means algorithm harder to converge and needs more iterations. K-means algorithm may never be able to converge if kmeansEta is too small.  Results: smaller kmeansEta increases the accuracy of the clusters if the algorithm is able to converge. |
| kmeansMaxIterations | Performance: larger kmeansMaxIterations gives the algorithm more chance to converge and leads to lower SSE. But also increases the algorithm running time if k-means is not able to meet the convergence requirement.  Results: Larger kmeansMaxIterations leads to more accurate clustering results with lower sum of squared distance. |

1. **Further discussion**

* K-Means: How to improve the efficiency?
* Design smart algorithm to sample initial centroids

Initial centroids quality can largely affect the quality and performance of k-means result

* Improve efficiency of k-means by restarting the algorithm when certain conditions is met
* Spark: How to speed up the processing?
* Allocate more cores for the computation.

When creating SparkConf, change local to local[\*]. Your computer will be able to allocate more cores for the task computations

Use local: 

Use local[\*]: 

We can see that local[\*] is much faster than simply using local (the time unit is millisecond)

* Cache frequently used RDD for computations.

e.g. vectors are used in each iteration of kmeans algorithm, so it is better to cache vectors

* Use reduceByKey to replace groupByKey if applicable

In reduceByKey(), pairs on the same machine with the same key are combined before the data is shuffled. Then the function is called again to reduce all the values from each partition to produce one final result.

In groupByKey(), all the key-value pairs are shuffled around. There is a lot of unnecessary data to be transferred over the network.