User Clustering

1 Cluster

In this section, we use spark to implement a k-means algorithm using Scala which clusters some posts, which are from some popular question-answer platforms, according to their score and domains. Each row of the input consists of `PostingType`, `ID`, `ParentID`, `Score`, `Domains`. Note that some of the five input elements may be empty.

1.1 Brief decription of K-mean method

Here is the solution flow. First, we group the posts by the questions. Because a question may have lots of answers. Then, we get the highest score of all the answers corresponding to each question. Then, we try to vectorize each question by defining (questioned*`DomainSpreed`, highest answer score). We can see that the parameter `DomainSpreed` can change how far each question is from each other in the feature space. Then, by setting the number of clusters, we can use K-mean cluster method by iteration for maximum `kmeansMaxIterations` times or until the change of Euclidean distance of the cluster centroid is smaller than `kmeansEta`. Here, we can see that the parameter `kmeansMaxIterations` and `kmeansEta` can affect the accuracy of the clustering.

1.2 Analysis of results

In this section, we will the result for 15 clusters, which equal to the size of topic domian. Here, we define `DomainSpreed` as 50000. Here, we mainly focus on the main topic of each resulting cluster and the percentage of the domain topic, median score and average score in each cluster.

Case 1 number of clusters = 15

topic	percentage	size	median score	average score
Data-Analysis	100%	364567	1	3
Machine-Learning	100%	366113	1	3
Architecture	100%	3053	1	3
Computer-Systems	100%	113982	1	3
Software-Engineering	100%	14488	1	2
Silicon Valley	100%	55409	1	4
Algorithm	100%	316259	1	2
Deep-learning	100%	95602	1	4
Compute-Science	100%	383958	1	3
Internet-Service-Providers	100%	24001	2	3
Security	100%	181528	2	4
Big-Data	100%	175005	2	4
Programming-Language	100%	13198	3	5
Embedded-System	100%	4093	3	5
Cloud-services	100%	10566	4	7

Table 1 Result of clustering

From Table 1, we can see that the percentages of questions of the domain topic in each cluster are all nearly 100% and there are 15 different clusters with different domain topic. Here, we can see the K-mean clustering works well under this parameter set up. Then, we will focus on the results data we get and try to explore for some insights of the question and answer forum.

From the results, we can get the following conclusions by analysis.

- "Computer science" is the most popular topic, whose size is 383958. "Data-Analysis", "Machine-Learning" and "Algorithm" are also popular, the size of which are all over 300000. "Architecture" is the rarest topic that people want to discuss.
- All the median score is obviously smaller than the average score, which means that most of the answer has fairly low scores and only a little number of them can be become the high score ones. This means that most of answers of the questions may give much help to people and only few answers have high quality.

1.3 Analysis of parameters of K-mean

In this part, we will explore how different parameters impact the performance and clustering results of k-means. From the description of the K-mean algorithm at the beginning of Section 2.1, we can already get the functions of parameters `kmeansKernels`, `DomainSpreed`, `kmeansMaxIterations` and `kmeansEta`. From the functions of each clusters, we can also analyze some impacts of these parameters.

1.3.1DomainSpreed

`DomainSpreed` is to control how far each question is from each other in the feature space. If the `DomainSpreed` is large, the nodes of questions are far enough to each other. Thus, it is easier for the K-mean clustering to cluster, which means that the percentage of the centroid's domain in the corresponding cluster are higher comparing to that of smaller `DomainSpreed`. We change the value of `DomainSpreed` as 5 and 50000. From the following results, we can see that the `DomainSpreed` can affects the performance of the clustering. When `DomainSpreed` is set too small, the components of each resulting cluster may not be very pure, which means that the performance of clustering become worse.

Topic (`DomainSpreed` = 50000)	Percentage (`DomainSpreed` = 50000)	Topic (`DomainSpreed` = 5)	Percentage (`DomainSpreed` = 5)
Data-Analysis	100%	Data-Analysis	100.0%
Machine-Learning	100%	Machine-Learning	100.0%
Architecture	100%	Machine-Learning	30.8 %
Computer-Systems	100%	Computer-Systems	69.4 %
Software-Engineering	100%	Software-Engineering	32.2 %
Silicon Valley	100%	Machine-Learning	47.9 %
Algorithm	100%	Algorithm	100.0%
Deep-learning	100%	Deep-learning	78.4 %
Compute-Science	100%	Compute-Science	97.5 %
Internet-Service-Providers	100%	Machine-Learning	23.1 %
Security	100%	Security	95.5 %
Big-Data	100%	Big-Data	96.4 %
Programming-Language	100%	Machine-Learning	57.1 %
Embedded-System	100%	Data-Analysis	21.6 %
Cloud-services	100%	Compute-Science	18.1 %

Table 2 Results for DomainSpreed = 50000 and 5

From Table 2, we can see that when `DomainSpreed` is set too small, the K-mean clustering may not work so well. Some rare topics miss in other popular topic and the questions of the same popular topics may be clustered into different clusters.

1.3.2 kmeansMaxIterations and kmeansEta

kmeansMaxIterations and kmeansEta control the converage criteria of the K-mean method. The impact of kmeansMaxIterations and kmeansEta is very easy to analyze.

- When kmeansMaxIterations is set too small and kmeansEta is set too large, the K-mean cluster may stop early before it coverage. The accuracy of the K-mean clustering may become poor.
- When kmeansMaxIterations is set too large and kmeansEta is set too small, the K-mean cluster may continue to run even when it can be treated as converged. This may waste lots of computing resources.

So, there are a tradeoff between accuracy and resource consumption when setting kmeansMaxIterations and kmeansEta.

1.3.3 kmeansKernels

kmeanKernels control the number of clusters.

- When the number of clusters is set small, many of elements of different clusters may be classified into the same clusters.
- When the number of cluster is set too large, the elements of the same cluster may be classified into different clusters, which make it difficult to get what the actual clustering is.

So, we need to set a suitable kmeansKernels based on the prior knowledge.

1.4 Further discussion on the system performance

The time for the program to the clustering results take about 1 min so far. There are some potential ways to improve and speed up the performance.

During the data processing, we create 7 RDDs and those RDDs. Because we only focus on the final results, we can use the pipeline to save the space of memory.

We use the join in groupedPostings(), which may cost lots of time. There maybe a better way to use ReduceByKey if it is possible.