## Programming Assignment: StackOverflow (2 week long assignment)

You have not submitted. You must earn 8/10 points to pass.

|  |  |
| --- | --- |
| **Deadline** | Pass this assignment by March 26, 11:59 PM PDT |

1. [Instructions](https://www.coursera.org/learn/scala-spark-big-data/programming/FWGnz/stackoverflow-2-week-long-assignment)
2. [My submission](https://www.coursera.org/learn/scala-spark-big-data/programming/FWGnz/stackoverflow-2-week-long-assignment/submission)
3. [Discussions](https://www.coursera.org/learn/scala-spark-big-data/programming/FWGnz/stackoverflow-2-week-long-assignment/discussions)

# StackOverflow

To start, first download the assignment: [stackoverflow.zip](http://alaska.epfl.ch/~dockermoocs/bigdata/stackoverflow.zip" \t "/home/hjmao/Documents\\x/_blank). For this assignment, you also need to download the data (170 MB):

[http://alaska.epfl.ch/~dockermoocs/bigdata/stackoverflow.csv](http://alaska.epfl.ch/~dockermoocs/bigdata/stackoverflow.csv" \t "/home/hjmao/Documents\\x/_blank)

and place it in the folder: src/main/resources/stackoverflow in your project directory.

The overall goal of this assignment is to implement a distributed k-means algorithm which clusters posts on the popular question-answer platform StackOverflow according to their score. Moreover, this clustering should be executed in parallel for different programming languages, and the results should be compared.

The motivation is as follows: StackOverflow is an important source of documentation. However, different user-provided answers may have very different ratings (based on user votes) based on their perceived value. Therefore, we would like to look at the distribution of questions and their answers. For example, how many highly-rated answers do StackOverflow users post, and how high are their scores? Are there big differences between higher-rated answers and lower-rated ones?

Finally, we are interested in comparing these distributions for different programming language communities. Differences in distributions could reflect differences in the availability of documentation. For example, StackOverflow could have better documentation for a certain library than that library's API documentation. However, to avoid invalid conclusions we will focus on the well-defined problem of clustering answers according to their scores.

Note: for this assignment, we assume you recall the K-means algorithm introduced during Parallel Programming part of the specialization. You may refer back to the [K-means assignment text](http://alaska.epfl.ch/~dockermoocs/bigdata/kmeans/kmeans.html" \t "/home/hjmao/Documents\\x/_blank) for an overview of the algorithm!

## The Data

You are given a CSV (comma-separated values) file with information about StackOverflow posts. Each line in the provided text file has the following format:

A short explanation of the comma-separated fields follows.

You will see the following code in the main class:

It corresponds to the following steps:

1. lines: the lines from the csv file as strings
2. raw: the raw Posting entries for each line
3. grouped: questions and answers grouped together
4. scored: questions and scores
5. vectors: pairs of (language, score) for each question

The first two methods are given to you. You will have to implement the rest.

## Data processing

We will now look at how you process the data before applying the kmeans algorithm.

### Grouping questions and answers

The first method you will have to implement is groupedPostings:

In the raw variable we have simple postings, either questions or answers, but in order to use the data we need to assemble them together. Questions are identified using a "postTypeId" == 1. Answers to a question with "id" == QID have (a) "postTypeId" == 2 and (b) "parentId" == QID.

Ideally, we want to obtain an RDD with the pairs of (Question, Iterable[Answer]). However, grouping on the question directly is expensive (can you imagine why?), so a better alternative is to match on the QID, thus producing an RDD[(QID, Iterable[(Question, Answer))].

To obtain this, in the groupedPostings method, first filter the questions and answers separately and then prepare them for a join operation by extracting the QID value in the first element of a tuple. Then, use one of the join operations (which one?) to obtain an RDD[(QID, (Question, Answer))]. Then, the last step is to obtain an RDD[(QID, Iterable[(Question, Answer)])]. How can you do that, what method do you use to group by the key of a pair RDD?

Finally, in the description we made QID, Question and Answer separate types, butin the implementation QID is an Int and both questions and answers are of type Posting. Therefore, the signature of groupedPostings is:

This should allow you to implement the groupedPostings method.

### Computing Scores

Second, implement the scoredPostings method, which should return an RDD containing pairs of (a) questions and (b) the score of the answer with the highest score (note: this does not have to be the answer marked as "acceptedAnswer"!). The type of this scored RDD is:

For example, the scored RDD should contain the following tuples:

Hint: use the provided answerHighScore given in scoredPostings.

### Creating vectors for clustering

Next, we prepare the input for the clustering algorithm. For this, we transform the scored RDD into a vectors RDD containing the vectors to be clustered. In our case, the vectors should be pairs with two components (in the listed order!):

* Index of the language (in the langs list) multiplied by the langSpread factor.
* The highest answer score (computed above).

The langSpread factor is provided (set to 50000). Basically, it makes sure posts about different programming languages have at least distance 50000 using the distance measure provided by the euclideanDist function. You will learn later what this distance means and why it is set to this value.

The type of the vectors RDD is as follows:

For example, the vectors RDD should contain the following tuples:

Implement this functionality in method vectorPostings and by using the given the firstLangInTag helper method.

(Idea for test: scored RDD should have 2121822 entries)

## Kmeans Clustering

Based on these initial means, and the provided variables converged method, implement the K-means algorithm by iteratively:

* pairing each vector with the index of the closest mean (its cluster);
* computing the new means by averaging the values of each cluster.

To implement these iterative steps, use the provided functions findClosest, averageVectors, and euclideanDistance.

**Note 1:**

In our tests, convergence is reached after 44 iterations (for langSpread=50000) and in 104 iterations (for langSpread=1), and for the first iterations the distance kept growing. Although it may look like something is wrong, this is the expected behavior. Having many remote points forces the kernels to shift quite a bit and with each shift the effects ripple to other kernels, which also move around, and so on. Be patient, in 44 iterations the distance will drop from over 100000 to 13, satisfying the convergence condition.

If you want to get the results faster, feel free to downsample the data (each iteration is faster, but it still takes around 40 steps to converge):

However, keep in mind that we will test your assignment on the full data set. So that means you can downsample for experimentation, but make sure your algorithm works on the full data set when you submit for grading.

**Note 2:**

The variable langSpread corresponds to how far away are languages from the clustering algorithm's point of view. For a value of 50000, the languages are too far away to be clustered together at all, resulting in a clustering that only takes scores into account for each language (similarly to partitioning the data across languages and then clustering based on the score). A more interesting (but less scientific) clustering occurs when langSpread is set to 1 (we can't set it to 0, as it loses language information completely), where we cluster according to the score. See which language dominates the top questions now?

## Computing Cluster Details

After the call to kmeans, we have the following code in method main:

Implement the clusterResults method, which, for each cluster, computes:

* (a) the dominant programming language in the cluster;
* (b) the percent of answers that belong to the dominant language;
* (c) the size of the cluster (the number of questions it contains);
* (d) the median of the highest answer scores.

Once this value is returned, it is printed on the screen by the printResults method.

## Questions

* Do you think that partitioning your data would help?
* Have you thought about persisting some of your data? Can you think of why persisting your data in memory may be helpful for this algorithm?
* Of the non-empty clusters, how many clusters have "Java" as their label (based on the majority of questions, see above)? Why?
* Only considering the "Java clusters", which clusters stand out and why?
* How are the "C# clusters" different compared to the "Java clusters"?