

Lab 6

In this lab, we continue our work on the Amazon dataset using Apache Spark. Your task is similar to that of Lab 3. Specifically, given the original Amazon food dataset (that you can find in the HDFS file system at `/data/students/bigdata-01QYD/Lab3/Reviews.csv`), find all the pairs of products frequently reviewed together.

In the following Task 1 you find the operations and analyses you are required to perform on the input dataset.

Task 1

The input Amazon food dataset (available in the HDFS shared folder of the BigData@Polito cluster: `/data/students/bigdata-01QYD/Lab3/Reviews.csv`) lists all the reviews per-row (one review per line), and it is comma-separated. In each line, two of the columns represent the user id and product id (third and second columns, respectively). The schema of `Reviews.csv` is the following:

```
Id,ProductId,UserId,ProfileName,HelpfulnessNumerator,HelpfulnessDenominator,Score,Time,Summary,Text
```

On the web site you can download the file `ReviewsSample.csv`. It contains a sample of `Reviews.csv`. You can use it to perform some initial tests.

Write a single Spark application that:

1. Transposes the original Amazon food dataset, obtaining a PairRDD in which there is one pair, structured as follows, for each user:
(`user_id`, list of the `product_ids` reviewed by `user_id`)

The returned PairRDD contains one pair for each user, which contains the `user_id` and the complete list of (distinct) products reviewed by that user. If user `user_id` reviewed more times the same product, that product must occur only one time in the returned list of the `product_ids` reviewed by `user_id`;

2. Counts the frequencies of all the pairs of products reviewed together (the frequency of a pair of products is given by the number of users who reviewed both products);
3. Writes on the output folder all the pairs of products that appear more than once and their frequencies. The pairs of products must be **sorted by decreasing frequency**.

Inspect the output of your application to search for interesting facts.

Pay attention that the line starting with "Id," is the header of the file and must not be considered.

On the web site you can download the file `ReviewsSample.csv`. It contains a sample of `Reviews.csv`. You can use it to perform some initial tests locally.

Task 2 - Bonus task

Extend the implemented application in order to write on the standard output the top 10, most frequent, pairs of products and their frequencies.

Note that Spark 1.6, or above, provides the following actions that can be applied on an RDD of type `JavaRDD<T>` (the same actions can also be applied on `PairRDDs`):

4. `List<T> top(int n, java.util.Comparator<T> comp)`
5. `List<T> takeOrdered (int n, java.util.Comparator<T> comp)`

top returns the **n largest elements** of the RDD based on the specified Comparator

takeOrdered returns **the n smallest elements** of the RDD based on the specified Comparator

Note that the standard output of the driver is stored in the log files of your application if you run `spark-submit` by setting `--deploy-mode cluster`. Use `--deploy-mode client` if you want to see the standard output of driver in the terminal of `jupyter.polito.it`.

Use the following steps to access the log files if you use `--deploy-mode cluster`.

How to access logs files

If you are connecting from outside Polito and you submit your application on the cluster by using `spark-submit` you can proceed as follows to retrieve the log files from the command line:

1. Open a Terminal on the gateway `jupyter.polito.it`
2. Execute the following command in the Terminal:

```
yarn logs -applicationId application_1521819176307_2195
```

The last parameter is the application/job ID. You can retrieve the job ID of your application on the HUE interface: <https://bigdatalab.polito.it/hue/jobbrowser#!/jobs>