Lab 2

In this lab, you will write by your own a complete Hadoop application. Start by importing the template project available in Lab2_Skeleton.zip. Once you have imported the template, modify the content of the classes to implement the application described in the following. The template contains the skeleton of a standard MapReduce application based on three classes: Driver, Mapper, and Reducer. Analyze the problem specification and **decide if you really need all classes** to solve the assigned problem.

From now on, keep in mind always (even if we do not explicitly ask for it) the complexity of your program. Specifically try to understand, what is the effort you will require to the cluster in terms of time, network and I/0

- How many pairs and bytes are read from HDFS?
- How many pairs and bytes are emitted by the mappers and hence how many data are sent on the network?

1. Filter an input dataset

If you completed Lab 1, you should now have (at least one) large files with the word frequencies in the Amazon food reviews, in the format `word\tnumber`, where number is an integer (a copy of the output of Lab 1 is available in the HDFS shared folder /data/students/bigdata-01QYD/Lab2/). You should also have realized that inspecting these results manually is not feasible. Your task is to write a Hadoop application to filter the content of the output of Lab 1 and analyze the filtered data.
The filter you should implement is the following:

- Keep only the lines containing words that start with “ho”

Store the selected lines (word\tnumber) in the output folder.

How large is the result of this filter? Do you need to filter more?

Modify the application to accept the beginning string as a command-line parameter.
Execute the new version of the program to select the words starting with the prefix that you prefer.

If you need to access the log files associated with the execution of your application, use the following commands in the terminal of jupyter.polito.it:

1. To retrieve the log associated with the standard output
   - yarn logs -applicationId <id of your application> -log_files stdout
     - The “id of your application” is printed on the terminal at the beginning of the execution of your application
       - The format of “id of your application” is application_number_number
       - Example of “application id” application_1584304411500_0009
     - You can retrieve the application id also from the HUE interface
   - The returned result contains one stdout log section for each task
     - One for the Driver
     - One for each Mapper
     - One for each Reducer

2. To retrieve the log associated with the standard error
   - yarn logs -applicationId <id of your application> -log_files stderr

2. Filter and count

Extend the previous application.

The new version of your application must:

1. Select only the lines containing words that start with the prefix provided as a parameter (like the previous application) and store them in the output folder.
2. **Print on the standard output of the Driver** also the number of selected words and the number of discarded words.

Bonus task

If you completed the bonus task of lab 1, try your filter on the 2-grams you have generated.
If you did not complete the bonus task of lab 1, you can use the files available in the HDFS shared folder /data/students/bigdata-01QYD/Lab2BonusTrack/

What is the size of this new input dataset, compared to the simple word counts (1-grams) we used in the previous step? Did you really need the cluster to filter 1-grams? What about 2-grams?

Implement a new application that selects all the 2-grams that contain, at any position, the word “like” (i.e., “like” can be either the first or the second word of the selected 2-grams). What do you think will be, most likely, the other word?