Lab 5

The objective of this laboratory is to start playing around with Apache Spark.

1. Problem specification

If you completed Lab 1, you should now have (at least one) files with the word frequencies in the Amazon food reviews, in the format `word\tfreq`, where *freq* is an integer (a copy of the output of Lab 1 is available in the HDFS shared folder /data/students/bigdata-01QYD/Lab2/). Your task is to write a **Spark** application to filter these results, analyze the filtered data and compute some statistics on them.

Task 1

The first filter you should implement is the following:
- Keep only the lines containing words that start with the prefix "ho"

The returned RDD contains the set of lines (`word\tfreq`) that satisfy the filtering operation.

Print on the standard output the following statistics, based on the content of the RDD returned by the filtering operation:
- The number of selected lines
- The maximum frequency (*maxfreq*) among the ones of the selected lines (i.e., the maximum value of *freq* in the lines obtained by applying the filter).

Task 2

Extend the previous application. Specifically, in the second part of your application, among the lines selected by the first filter, you have to apply another filter to select only the most frequent words. Specifically, your application must select those lines that contain words with a frequency (*freq*) greater than 80% of the maximum frequency (*maxfreq*) computed before.

Hence, implement the following filter:
- Keep only the lines with a frequency *freq* greater than 0.8*maxfreq.

Finally, perform the following operations on the selected lines (the ones selected by applying both filters):
- Count the number of selected lines and print this number on the standard output
- Save the selected words (without frequency) in an output folder (one word per line)

2. Testing the application

Run your application

1. Create a Jupyter notebook (select PySpark (Local)) and run you application on the input HDFS folder /data/students/bigdata-01QYD/Lab2/. Set the name of the output folder in your code.
    a. Analyze the returned results (the statistics/results printed on the standard output and the content of the output folder)

2. Create a Python script and execute it with the spark-submit command.
   a. Note. In this version of the code prefix, input folder, and output folder must be specified by means of three command line arguments (hint: use sys.argv[])
   b. Analyze the returned results. They should be consistent with the previous ones.
   c. Run your Python script two times:
      i. Run it on the gateway by using the --master local option of spark-submit
      ii. Run it on the nodes of the cluster by using the --master yarn option of spark-submit

How to run your application

- Approach based on Jupyter notebooks
  - Pyspark (Local) notebook - To run your application on the gateway
    - Open a browser and connect to jupyter.polito.it
    - Log in and open a "Pyspark (local)" notebook
    - Write your application in the notebook and run it on the gateway (data are read from and stored on HDFS but driver and executors are instantiated on the gateway)

  - PySpark (Yarn) notebook - To run your application on the nodes of the cluster
    - Open a browser and connect to jupyter.polito.it
    - Log in and open a "Pyspark (Yarn)" notebook
    - Write your application in the notebook and run it on the nodes of the cluster (data are read from and stored on HDFS and driver and executors are instantiated on the nodes/servers of the cluster BigData@Polito)

- Approach based on a "standalone" python script (a textual file with the extension .py) and the spark-submit command
  - spark-submit
    - Write your Python application by using your preferred editor and save it in a Python file (.py)
      - To run a Spark script with spark-submit, your script must explicitly create the SparkContext object. To do this, insert the following lines of code at the beginning of your script

        from pyspark import SparkConf, SparkContext
        conf = SparkConf().setAppName("Name of my application")
        sc = SparkContext(conf = conf)

      - Remember also to stop/close the SparkContext at the end of your application

        sc.stop()

    - Open a browser and connect to jupyter.polito.it
    - Log in and open a Terminal
    - Copy your Python script in the local file system of the gateway (use the drag and drop approach)
    - Run your Python script by executing the following command in the terminal
      spark-submit --master yarn --deploy-mode client <your_script>

- Note that <your_script> must end by .py for Spark to interpret it as Python application
- The driver is instantiated on the gateway (--deploy-mode client) and the executors on the nodes of the cluster (--master yarn)
- You can use --master local if you want to run also the executors of your application on the gateway (data are read from and stored on HDFS but driver and executors are instantiated on the gateway with the setting --master local --deploy-mode client)

3. [Optional] Analyze DAGs and statistics of the submitted jobs

The following steps work only if you are using the LABINF's PC. md1-bigdata.polito.it is not accessible from outside Polito's network and it is blocked by Polito's wifi.

1. Connect to https://md1-bigdata.polito.it:18489/
2. Search your jobs (type your username in the search box to select your jobs)
3. Click on the job you want to analyze and navigate the tabs to see the DAGs and statistics of your job