Lab 11

In this lab, we analyze a stream of tweets to identify appearing hashtags and compute the number of occurrences of each over a sliding window. Your task is to extract the hashtags from the input tweets and compute their occurrences every 10 seconds, using the last 30 seconds of data.
The stream of tweet data is simulated by uploading a set of files to an input folder during the execution of your application. Each input file contains one tweet per line. Each line of the input files has the following format:

- *userId***\t***text_of_the_tweet*
    - The two fields are separated by a tab

For example, the line

26976263**\t**Gym time!!!! #fitness

means that user **26976263** tweeted the text "**Gym time!!!! #fitness**". The text of this tweet also contains a hashtag: **#fitness**

Ex. 1

Write a Spark streaming application, based on DStreams, that counts the number of occurrences of each hashtag appearing in the input data streaming. Specifically, **every 10 seconds**, your application must

- Extract the hashtags appearing in **the last 30 seconds of the input data stream**,
- Count the number of occurrences of each (extracted) hashtag by considering only **the last 30 seconds of data** (i.e., the last 30 seconds of data of the input data stream)
- Store in the output folders, sorted by number of occurrences, the pairs (number of occurrences, hashtag) related to **the last 30 seconds of data**,
- Print on the standard output the first 10 hashtags in terms of number of occurrences, related to **the last 30 seconds of data.**

The application performs the analysis every 10 seconds, using the last 30 seconds of streaming data (i.e., window length = 30 seconds and sliding interval = 10 seconds).

The input data stream is based on the content of an input folder in which, during the execution of the application, you will upload files formatted according to the format specified in the first part of this problem specification. Upload the input files one at a time to simulate a stream of data. Specifically:

1. Create a first folder on the file system of jupyter.polito.it and upload the files you want to use in that folder
2. Create a second empty folder (the folder associated with the input of your application)
3. Copy, one at a time, the input files from the first input folder to the second input folder of your application by using the command line **cp**
    E.g., cp first_folder/tweets_blockaa second _folder/

4. Pay attention that if a file is already in the second folder (the input folder of your application) and you copy another version of the same file, the system will not consider the new version of the file
5. Pay attention to **stop your streaming context after your tests** by invoking
   ssc.stop(stopSparkContext=False)

Some example input files are available on the website of the course (Lab11Data.zip)

Note. If you run this application locally on your PC:

1. Create the input files in a folder and then copy them to the input folder of your application (one file at a time to simulate the stream of data)
2. Copy the files in the input folder of your application by using the command line **cp**
   E.g., cp tweets_blockaa input_folder/
3. **Do not use the graphical interface to copy the file in the input folder of your application. Otherwise, the output of your application will be empty**
4. If you update the content of a file that is already in the input folder of your application, the updated version of the file will not be considered by the Spark streaming engine
5. sortBy cannot be directly applied to DStreams. You must use transform().

Ex. 2

We are interested in implementing a simple alert system that prints only “relevant” hashtags to standard output. A hashtag is defined as relevant if it occurred at least 100 times in the last 30 seconds.
Extend your application to print to standard output and write to the output folders only the hashtags that occurred at least 100 times in the last 30 seconds. The returned hashtags must be sorted by number of occurrences.
Analogously to the first part, this application must perform the analysis every 10 seconds, using the last 30 seconds of streaming data (i.e., window length = 30 seconds and sliding interval = 10 seconds).

Shut down JupyterHub container

As soon as you complete all the tasks and activities on JupyterHub environment, please remember to shut down the container to let all your colleagues in all the sessions connect on JupyterHub and do all the lab activities.

1. Go into File -> Hub Control Panel menu
2. A new browser tab opens with the “Stop My Server” button. Click on it and wait till it disappears.