Project Apache Spark

This project can be done in teams of 2 or 3 students (or individually if you really prefer), and has to be sent via eCampus https://ecampus.paris-saclay.fr/mod/assign/view.php?id=1149991 by **Tuesday, May 2nd 2023, 20h00 CEST**.

The project should be submitted as a zip archive file with a name of the form: `spark23-name1-name2.zip` (replace name1, name2 by your lastnames). It should contain at least a report on the project (pdf), the source code files (and instructions on how to run the code), and may contain any other file that you may find necessary. Only one submission per team is required and expected.

> The project is kept intentionally open-ended to allow you some creativity – in the context of the general lines outlined below. Hence, you are reasonably free to organize the pipeline, but make sure we can easily run your code.

Processing API Data as a Stream

1. Identify a streaming API that publishes some textual data (for instance we used to suggest Twitter, but it appears the free version has been discontinued. You may consider other social networks such as twitch, news apis, or governmental administrations like PRIM for mobility network).

   > Keep in mind that many of those API require some access key or token provided only after the user registration, and that registration can be manually vetted and therefore take quite some time. So do it as soon as possible.

2. Create a Python server that will connect to your API and send individual "messages" and their metadata (dates...) to a socket. You may use for instance `socket`.

3. Read the data in Spark Streaming, and tag the data to perform sentiment analysis (you may use `textblob`).

   > A quickstart on how to use `textblob` is available on the documentation site https://textblob.readthedocs.io/en/dev/quickstart.html; by default, you do not need to use / know much more than this.

   > Obviously, we could add the sentiment tags directly in the listener, but for the sake of the exercise we will use Spark.

Analyzing using Spark

4. In Spark Streaming perform some analysis that involves a ML algorithm and a window operation to extract interesting insights from your data stream. For instance, you can:

   - cluster the messages according to sentiments and their location, then,
   - evaluate the cluster size in sliding windows.

   > MLLib has some implementations of clustering algorithms on streams, such as streaming k-means: https://spark.apache.org/docs/latest/mllib-clustering.html#streaming-k-means
   > The window operation is part of the transformation operations of Spark Streaming, and is discussed here: https://spark.apache.org/docs/latest/streaming-programming-guide.html#window-operations

Visualization

5. Create a dynamic dashboard from your stream (i.e., that can be refreshed, not just plotting some constant data).

   > *Suggestions* To visualize data, the simplest way would be to use a Jupyter notebook, and use the `matplotlib` library to plot the data, `seaborn` or `plotly`.