BIG DATA PROCESSING

ASSIGNMENT 2: SPARK STREAMING,
SPARK STRUCTURED STREAMING AND
ADDITIONAL SPARK LIBRARIES.

BACKGROUND.

It's been already a few weeks since you started your short-term internship in the Big Data Engineering Department of the start-up OptimiseYourJourney, which will enter the market next year with a clear goal in mind: "*leverage Big Data technologies for improving the user experience in transportation*". Your contribution in Assignment 1 has proven the potential OptimiseYourJourney can obtain by using Spark Core and Spark SQL to analyse public transportation datasets as **Dublin Bus GPS sample data from Dublin City Council (Insight Project)**: https://data.gov.ie/dataset/dublin-bus-gps-sample-data-from-dublin-city-council-insight-project

OptimiseYourJourney

In the department meeting that has just finished your boss was particularly happy, again.

- First, she thinks this very same Dublin Bus dataset provides a great opportunity to explore the potential of Spark Streaming and Spark Structured Streaming in performing real-time data analysis. To do so, she asks you to adapt the exercises of the previous assignment for their application to these new technologies.
- Second, she is curious about the possibilities other Spark libraries, specifically devoted to Graph and Machine Learning algorithms, would offer when applied to this Dublin Bus Dataset. To do so, she asks you to write a short report about it.

DATASET.

Each real-time dataset you will be dealing with contains a number of files <'f1.csv', 'f2.csv', 'f3.csv', fn.csv'> and represents n batches (of 1 file each) arriving over time for their real-time data analysis. The datasets are provided to you in the folder my_dataset of **Canvas => 5_Assignments => A02.zip**.

As in Assignment 1, each row of an Assignment 2 dataset file contains the following fields:
Date , Bus_Line , Bus_Line_Pattern , Congestion , Longitude , Latitude , Delay , Vehicle , Closer_Stop , At_Stop

- **(00) Date**
  - A String representing the date of the measurement with format <%Y-%m-%d %H:%M:%S>
  - Example: "2013-01-01 13:00:02"
- **(01) Bus_Line**
  - An Integer representing the bus line.
  - Example: 120
- **(02) Bus_Line_Pattern**
  - A String identifier for the sequence of stops scheduled in the bus line (different buses of the same bus line can follow different sequence of stops in different iterations).
  - Example: "027B1001" (it can also be empty "").
- **(03) Congestion**
  - An Integer representing whether the bus is at a traffic jam (No => 0 / Yes => 1) .
  - Example: 0
- **(04) Longitude**
  - A Float representing the longitude position of the bus.
  - Example: -6.269634
- **(05) Latitude**
  - A Float representing the latitude position of the bus.
  - Example: 53.360504
- **(06) Delay**
  - An integer representing the delay of the bus with respect to its schedule (measured in seconds). It is a negative value if the bus is ahead of schedule.
  - Example: 90.
- **(07) Vehicle**
  - An integer identifier for the bus vehicle.
  - Example: 33304.
- **(08) Closer_Stop**
  - An integer identifier for the closest bus stop.
  - Example: 7486.
- **(09) At_Stop_Stop**
  - An integer representing whether the bus vehicle is at the bus stop right now (i.e., stopping at it for passengers to hop on / hop off). (No -> 0 and Yes -> 1)
  - Example: 0.

<u>TASKS / EXERCISES.</u>

The tasks / exercises to be completed as part of the assignment are described in the next pages of this PDF document.

- The following exercises are placed in the folder **my_code:**
  1. A02_ex1_spark_streaming.py
  2. A02_ex1_spark_structured_streaming.py
  3. A02_ex2_spark_streaming.py
  4. A02_ex2_spark_structured_streaming.py
  5. A02_ex3_spark_streaming.py
  6. A02_ex3_spark_structured_streaming.py

  - **Each exercise is worth 12.5 marks.**

  <u>Rules:</u>

  - On each exercise, your task is to complete the function **my_model** of the Python program. This function is in charge of specifying the Spark Job performing the real-time data analysis.
  - When programming my_model, you can create and call as many auxiliary functions as you need.
  - Do not alter any of the other functions provided: get_source_dir_file_names, streaming_simulation, create_ssc and my_main.
  - Do not alter the parameters passed to the function my_model.
  - The entire work must be done "within Spark":
    - The function my_model must start with a creation operation textFileStream or readStream. These operations track the arrival of the batch files and load their content to Spark Streaming and Spark Structured Streaming, respectively.
    - The function my_model must finish with an action operation pprint or writeStream printing by the screen the result of the Spark Streaming / Spark Structured Streaming job.
    - The function my_model must not contain any other action operation other than the pprint or writeStream appearing at the very end of the function.

- The following exercises are placed in the folder **my_report:**
  7. A02_report_additional_spark_libraries.docx

  - **The report is worth 25 marks.**
  - The report must contain a maximum of 1,000 words.

RUBRIC.

Exercises 1-6.

- 20% of the marks => Complete attempt of the exercise (even if it does not lead to the right solution or right format due to small differences).
- 40% of the marks => Right solution and format (following the aforementioned rules) for the "Small Dataset".
- 40% of the marks => Right solution and format (following the aforementioned rules) for any "Additional Dataset" test case we will generate. The marks will be allocated in a per test basis (i.e., if 4 extra test are tried, each of them will represent 10% of the marks).

Exercise 7.

- 25% of the marks => Originality.
- 25% of the marks => Relevance.
- 50% of the marks => Viability.

TEST YOUR SOLUTIONS.

- The folder **my_results/check_results** allows you to see if your code is producing the expected output or not.
  - The files **test_checker_spark_streaming.py** (resp. **test_checker_spark_structured_streaming.py**) need two files and compare whether their content is equal or not.
  - When you have completed one exercise (e.g., A02_ex1_spark_streaming.py), create a file with your result in the folder **my_results/check_results/Student_Solutions/**
  - Open the file **test_checker_spark_streaming.py** (resp. **test_checker_spark_structured_streaming.py** if the exercise you completed was A02_ex*_spark_structured_streaming.py) and edit the lines 142 and 141 (resp. 213 and 212) with the names of your file and the one you are comparing it against.
  - Run the program **test_checker_spark_streaming.py** (resp. **test_checker_spark_structured_streaming.py**).
    It will tell you whether your output is correct or not.

Main Message

Use the program **test_checker_spark_streaming.py** and **test_checker_spark_structured_streaming.py** to ensure that all your exercises produce the expected output (and in the right format!).

<u>SUBMISSION DETAILS / SUBMISSION CODE OF CONDUCT.</u>

Submit to Canvas by the 27th of November, 11:59pm.

- Submissions up to 1 week late will have 10 marks deducted.
- Submissions up to 2 weeks late will have 20 marks deducted.

On submitting the assignment you adhere to the following declaration of authorship. If you have any doubt regarding the plagiarism policy discussed at the beginning of the semester do not hesitate in contacting me.

Declaration of Authorship

I, ___YOUR NAME___, declare that the work presented in this assignment titled 'Assignment 2: Spark Streaming, Spark Structured Streaming & Additional Spark Libraries' is my own. I confirm that:

- This work was done wholly by me as part of my Msc. in Artificial Intelligence, my Msc. in Software Architecture and Design, my MSc. in Cloud Computing or my Msc. in Cyber Security at Munster Technological University.
- Where I have consulted the published work and source code of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this assignment source code and report is entirely my own work.

EXERCISE 1.

In Assignment 1, Exercise 1 was originally applied on Spark Core and Spark SQL. Now, in Assignment 2, we adapt the exercise to a real-time context for its application on Spark Streaming and Spark Structured Streaming. A full description of the original Exercise 1 can be found in the PDF description of Assignment 1 (pages 7-10). In the next sections:

- We provide the formal definition for the real-time version of the exercise.
    - We include any potential variation w.r.t. the original version of the problem due to limitations in the functionality of Spark Streaming and Spark Structured Streaming.
- We provide an example of a real-time dataset and the solution to be computed for it by Spark Streaming and Spark Structured Streaming.

EXERCISE: FORMAL DEFINITION

Given a program passing by parameters:

- The bus stop "bus_stop" (e.g., 279)

- The bus line "bus_line" (e.g., 40)

- A list of hours "hours_list" (e.g., ["08", "09"] for the intervals [8am - 9am) and [9am - 10), resp.)

and a dataset

- Containing several batches/files arriving over time (e.g., < f1.csv, f2.csv, f3.csv, f4.csv >)

your task is to:

- Compute the batch accumulated average delay of "bus_line" vehicles when stopping at "bus_stop" for each hour of "hours_list" during weekdays (you must discard any measurement taking place on a Saturday or Sunday).

The format of the solution computed after each new batch is processed must be:

- < (H1, AD1), (H2, AD2), ..., (Hn, ADn) >

where:

- Hi is one of the items of "hours_list". E.g., [8am - 9am).
- ADi is the average delay for all "bus_line" vehicles stopping at "bus_stop" within that hour on a weekday.
- Spark Streaming:
    - < (H1, AD1), (H2, AD2), ..., (Hn, ADn) > are also sorted by increasing order of ADi.
- Spark Structured Streaming:
    - < (H1, AD1), (H2, AD2), ..., (Hn, ADn) > are not sorted in any particular order.

<u>EXAMPLE – SMALL DATASET</u>

The folder “my_dataset/A02_ex1_micro_dataset” contains an example dataset containing 4 batches/ files

- < f1.csv, f2.csv, f3.csv, f4.csv >.

Given the aforementioned dataset and program parameters

- bus_stop = 279
- bus_line = 40
- hours_list = ["08", "09"]

the files of the folder “my_result”:

- A02_ex1_spark_streaming.txt => Solution for Spark Streaming
- A02_ex1_spark_structured_streaming.txt => Solution for Spark Structured Streaming

Once again, please note the <u>batch accumulated</u> average delay:

- Results for Batch 1 contain the average delay for all measurements of < f1.csv >.
- Results for Batch 2 contain the average delay for all measurements of < f1.csv, f2.csv >.
- Results for Batch 3 contain the average delay for all measurements of < f1.csv, f2.csv, f3.csv >.
- Results for Batch 4 contain the average delay for all measurements of < f1.csv, f2.csv, f3.csv, f4.csv >.

EXERCISE 2.

In Assignment 1, Exercise 2 was originally applied on Spark Core and Spark SQL. Now, in Assignment 2, we adapt the exercise to a real-time context for its application on Spark Streaming and Spark Structured Streaming. A full description of the original Exercise 2 can be found in the PDF description of Assignment 1 (pages 11-13). In the next sections:

- We provide the formal definition for the real-time version of the exercise.
    - We include any potential variation w.r.t. the original version of the problem due to limitations in the functionality of Spark Streaming and Spark Structured Streaming.
- We provide an example of a real-time dataset and the solution to be computed for it by Spark Streaming and Spark Structured Streaming.

EXERCISE: FORMAL DEFINITION

Given a program passing by parameters:

- The bus vehicle "vehicle_id" (e.g., 33145)

and a dataset

- Containing several batches/files arriving over time (e.g., < f1.csv, f2.csv, f3.csv, f4.csv >)

your task is to:

- Treat each batch individually (independent from the other batches) and compute per individual batch:

- Spark Streaming: The day the "vehicle_id" is serving the highest amount of different bus lines, and the sorted IDs of such bus lines => Exactly the same as in Assignment 1.
- Spark Structured Streaming: Per each day (irrespectively of whether it is the day serving the highest amount of bus lines or not), compute the number of bus lines the vehicle_id is serving and the sorted IDs of such bus lines => Different from what we did in Assignment 1.

The format of the solution computed must be:

- < (D1, LB1), (D2, LB2), ..., (Dn, LBn) >

where:

- Spark Streaming:
    - Di is the day of the month serving maximum amount of bus lines.
    - LBi is the list of bus lines served in that concrete day Di.
    - LBi is sorted by increasing order in the bus line IDs.
    - Note that < (D1, LB1), (D2, LB2), ..., (Dn, LBn) > will have as many pairs (Di, LBi) as days the bus vehicle is serving the very same amount of max bus lines.

    - If < (D1, LB1), (D2, LB2), ..., (Dn, LBn) > contains more than one pair (Di, LBi), then the pairs are also sorted by increasing order of Di.

- Spark Structured Streaming:
    - The format is actually < (D1, LB1, NB1), (D2, LB2, NB2), ..., (Dn, Lbn, NBn) >
    - There is a tuple (Di, LBi, Nbi) per day registering a measurement.
    - NBi is the number of bus lines served in that concrete day Di.
    - LBi is the list of bus lines served in that concrete day Di.
    - LBi is sorted by increasing order in the bus line IDs.
    - If < (D1, LB1, NB1), (D2, LB2, NB2), ..., (Dn, Lbn, NBn) > contains more than one tuple, then the pairs are not sorted sorted in any particular order.

EXAMPLE – SMALL DATASET

The folder “my_dataset/A02_ex2_micro_dataset” contains an example dataset containing 4 batches/ files

- < f1.csv, f2.csv, f3.csv, f4.csv >.

Given the aforementioned dataset and program parameters

- vehicle_id = 33145

the files of the folder “my_result”:

- A02_ex2_spark_streaming.txt => Solution for Spark Streaming
- A02_ex2_spark_structured_streaming.txt => Solution for Spark Structured Streaming

Once again, please note each batch is treated individually (independent from the other batches):

- Results for Batch 1 contain the bus lines of < f1.csv >.
- Results for Batch 2 contain the bus lines of < f2.csv >.
- Results for Batch 3 contain the bus lines of < f3.csv >.
- Results for Batch 4 contain the bus lines of < f4.csv >.

EXERCISE 3.

In Assignment 1, Exercise 4 was originally applied on Spark Core and Spark SQL. Now, in Assignment 2, we adapt the exercise to a real-time context for its application on Spark Streaming and Spark Structured Streaming. A full description of the original Exercise 3 can be found in the PDF description of Assignment 1 (pages 14-18). In the next sections:

- We provide the formal definition for the real-time version of the exercise.
    - We include any potential variation w.r.t. the original version of the problem due to limitations in the functionality of Spark Streaming and Spark Structured Streaming.
- We provide an example of a real-time dataset and the solution to be computed for it by Spark Streaming and Spark Structured Streaming.

EXERCISE: FORMAL DEFINITION

Given a program passing by parameters:

- The current time "current_time" (e.g., "2013-01-10 08:59:59")

- A bus stop "current_stop" (e.g., 1935)

- The time you allocate to yourself before you start walking again "seconds_horizon" (e.g., 1800 seconds, which is half an hour).

and a dataset

- Containing several batches/files arriving over time (e.g., < f1.csv, f2.csv, f3.csv, f4.csv >)

your task is to:

- Treat each batch individually (independent from the other batches) and compute per individual batch:

- Spark Streaming: Compute the first bus stopping at "current_stop" and the list of bus stops it bring us within that time "seconds_horizon" => Exactly the same as in Assignment 1.
- Spark Structured Streaming: Per each “vehicle_id” (irrespectively of whether it is the first one stopping at “current_stop” or not; moreover, irrespectively of whether it stops at “current_stop” or not) compute the time and stops it stops at. => Different from what we did in Assignment 1.

The format of the solution computed must be:

- < (V, [(T1, S1), (T2, S2), ..., (Tn, Sn)]) >

where:

- Spark Streaming:
    - V is the first bus vehicle stopping at "current_stop" after "current_time".
    - [(T1, S1), (T2, S2), ..., (Tn, Sn)] is the list of other bus stops this bus vehicle stops at before the end of "current_time" + "seconds_horizon", with Si being a bus stop and Ti the time stopping at it.

    - The list includes "current_stop" as S1, so as to know its associated value T1 (i.e., the time we hopped on the bus).

- Spark Structured Streaming:

    - The format is actually < (V1,T1, S1), (V1,T2, S2), ..., (V1,Tn1, Sn1), (V2,T1, S1), ..., (V2,Tn2, Sn2), ... , (Vk,Tnk, Snk) >
    - There is a tuple (V1,T1, S1) per "vehicle_id" and bus stop it stops at during the time horizon.
    - The measurements of each vehicle (V1,T1, S1), (V1,T2, S2), ..., (V1,Tn1, Sn1) are sorted in increasing order in the time the vehicle_id stops at the bus stops.

EXAMPLE – SMALL DATASET

The folder "my_dataset/A02_ex4_micro_dataset" contains an example dataset containing 4 batches/ files

- < f1.csv, f2.csv, f3.csv, f4.csv >.

Given the aforementioned dataset and program parameters

- current_time = "2013-01-10 08:59:59"
- current_stop = 1935
- seconds_horizon = 1800

the files of the folder "my_result":

- A02_ex4_spark_streaming.txt => Solution for Spark Streaming
- A02_ex4_spark_structured_streaming.txt => Solution for Spark Structured Streaming

Once again, please note each batch is treated individually (independent from the other batches):

- Results for Batch 1 contain the bus lines of < f1.csv >.
- Results for Batch 2 contain the bus lines of < f2.csv >.
- Results for Batch 3 contain the bus lines of < f3.csv >.
- Results for Batch 4 contain the bus lines of < f4.csv >.

EXERCISE 4.

Besides the Spark libraries covered in this semester < Spark Core, Spark SQL and Spark Real-Time Libs (Spark Streaming & Spark Structured Streaming) > Spark has also:

- A library especifially devoted to Graph algorithms
  - Spark GraphX (for working with RDDs)
  - Spark GraphFrames (for working with DataFrames)
- A library especifially devoted to Machine Learning algorithms
  - Spark MLlib (for working with RDDs)
  - Spark ML (for working with DataFrames)

Write a report of up to 1,000 words where you present and discuss:

- A novel exercise to be included in the data analysis of the Dublin Bus dataset involving the Spark Graph and/or Machine Learning libraries.

There is no need to implement the new exercise, you just need to discuss it in terms of:

- Its originality - It has to be different from the 3 exercises proposed in Assignments 1 and 2.
- Its relevance - Include a potential use-case derived from the exercise you are proposing.
- Its viability:
  - Do not implement the exercise, but briefly discuss in natural language (English and/or psudocode) the main steps that would be needed so as to implement it.
  - Include in the discussion whether, if you had to implement it, you would choose to implement it using the library version for working with RDDs or DataFrames. Justify your selection.
  - Position the new exercise in terms of difficulty with respect to the other four exercises proposed in this assignment.