INF 553 – Spring 2018

Assignment 5 Streaming Data

Deadline: 04/23 2018 11:59 PM PST

Assignment Overview

In this assignment we're going to implement some streaming algorithms. One is some analysis of Twitter stream. The other runs on a simulated data stream.

Environment Requirements

Python: 2.7 Scala: 2.11 Spark: 2.2.1
IMPORTANT: We will use these versions to compile and test your code. If you use other versions, there will be a 20% penalty since we will not be able to grade it automatically.
You can only use Spark RDD.

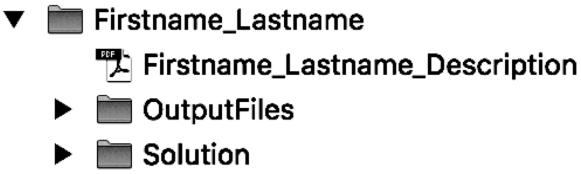
Write your own code!

For this assignment to be an effective learning experience, you must write your own code! I emphasize this point because you will be able to find Python implementations of most or perhaps even all of the required functions on the web. Please do not look for or at any such code! **Do not share code with other students in the class!!**

Submission Details

For this assignment you will need to turn in a Python, Java, or Scala program depending on your language of preference.

Your submission must be a .zip file with name: **<Firstname>_<Lastname>_hw5.zip**. The structure of your submission should be identical as shown below. The Firstname_Lastname_Description.pdf file contains helpful instructions on how to run your code along with other necessary information as described in the following sections. The *OutputFiles* directory contains the deliverable output files for each problem and the *Solution* directory contains your source code and .jar file.



```
▼ Firstname_Lastname
      Firstname_Lastname_Description
  ▶ OutputFiles
  ▶ Solution
```

Datasets

For Task2, you can download one file from Blackboard:

1. *userID.txt*
2. *StreamingSimulation.scala*

Task1: Twitter Streaming (50%)

You will use Twitter API and Spark streaming to track the popular hash tags on tweets and calculate the average length of tweets. For tracking popular hash tags, count them up over a 2-minute window sliding every 2 seconds.

Set up

- Creating credentials for Twitter APIs

In order to get tweets from Twitter, register on https://apps.twitter.com/ by clicking on "Create new app" and then fill the form click on "Create your Twitter app."
Second, go to the newly created app and open the "Keys and Access Tokens" tab. Then click on "Generate my access token." You will need to set these tokens as arguments when executing the code.

- Library dependencies

You can use "spark-streaming-twitter" (http://bahir.apache.org/docs/spark/current/spark-streaming-twitter/) and "spark-streaming" for this task, version 2.2.0 is recommended.

Execution Example

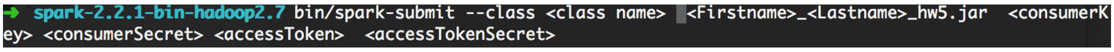
The arguments passed to your program are as follows:
consumerKey, consumerSecret, accessToken, accessTokenSecret

Following we present examples of how you can run your program with spark-submit both when your application is a Java/Scala program or a Python script.

A. **Example of running a Java/Scala application with spark-submit:**

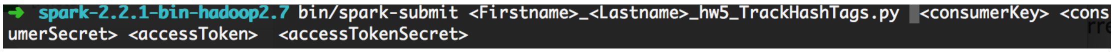
Notice that the argument class of the spark-submit specifies the main class of your n

You should use **TrackHashTags/AveTweetLength** as your class name for this task.

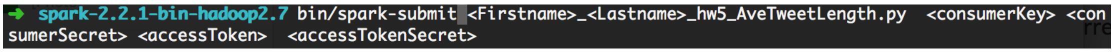


```
➜ spark-2.2.1-bin-hadoop2.7 bin/spark-submit --class <class name> <Firstname>_<Lastname>_hw5.jar  <consumerK
ey> <consumerSecret> <accessToken>  <accessTokenSecret>
```

B. **Example of running a Python application with spark-submit:**



```
➜ spark-2.2.1-bin-hadoop2.7 bin/spark-submit <Firstname>_<Lastname>_hw5_TrackHashTags.py <consumerKey> <cons
umerSecret> <accessToken>  <accessTokenSecret>
```



```
➜ spark-2.2.1-bin-hadoop2.7 bin/spark-submit <Firstname>_<Lastname>_hw5_AveTweetLength.py  <consumerKey> <con
sumerSecret> <accessToken>  <accessTokenSecret>
```

Result format:

For tracking popular hash tags, print the top 5 popular hash tags and the counts every 2 seconds, the format is like "#hashtag, count: 1".
For average tweets length, print the total number of tweets and the average length every 2 seconds, The format is like "Total tweets: 1, Average length: 1".

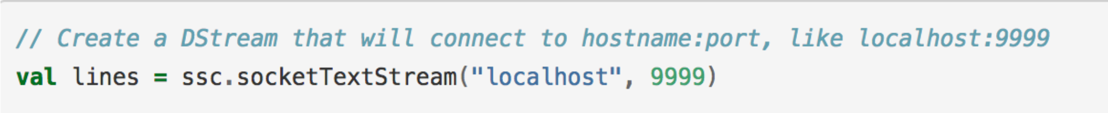
Task2: Flajolet-Martin algorithm (50%)

You are required to implement Flajolet-Martin algorithm to estimate the number of unique users in the stream.

Please find the *StreamingSimulation.scala* file under /data directory and compile it into *StreamingSimulation.jar*. Before you start running your Spark Stream code, run this jar file with command in the terminal: (This is just a simulation of data stream. It's not necessary to use Scala for Spark programming. Python is fine.)

java -cp <Path of StreamingSimulation.jar> StreamingSimulation <Path of userId.txt> 9999 100

In your Spark Stream code, please use this method to connect to the data stream that you created using the above command line:



```
// Create a DStream that will connect to hostname:port, like localhost:9999
val lines = ssc.socketTextStream("localhost", 9999)
```

The first argument is the host name and the second one the port number, which is 9999 in this case.

More guide of Spark Stream please refer to this: https://spark.apache.org/docs/latest/streaming-programming-guide.html

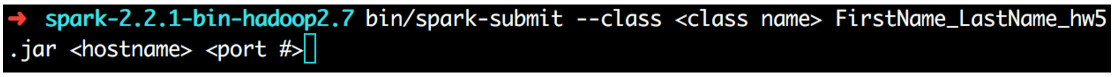
Execution Example

The first argument passed to your program (in the below execution) is the hostname of your computer. The second input is port number. Following we present examples of how you can run your program with spark-submit both when your application is a Java/Scala program or a Python script.

A. **Example of running a Java/Scala application with spark-submit:**

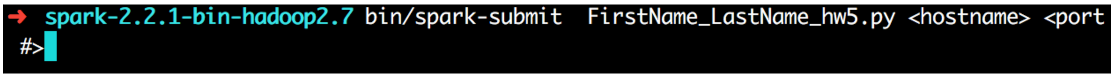
   Notice that the argument class of the spark-submit specifies the main class of your application and it is followed by the jar file of the application.

   You should use **UniqueUserCount** as your class name for the task.



```
➜  spark-2.2.1-bin-hadoop2.7 bin/spark-submit --class <class name> FirstName_LastName_hw5
.jar <hostname> <port #>
```

B. **Example of running a Python application with spark-submit:**



```
➜  spark-2.2.1-bin-hadoop2.7 bin/spark-submit  FirstName_LastName_hw5.py <hostname> <port
 #>
```

Result format:

Print the estimated number of unique users to the screen

Description File

Please include the following content in your description file:

1. Mention the Spark version and Python version
2. Describe how to run your program for both tasks

Submission Details

Your submission must be a .zip file with name: *<Firstname>_<Lastname>_hw5.zip*

Please include all the files in the right directory as following:

1. A description file: *<Firstname>_<Lastname>_desription.pdf*
2. All Scala scripts:
   <Firstname>_<Lastname>_task1_ TrackHashTags.scala
   <Firstname>_<Lastname>_task2_ AveTweetLength.scala
3. A jar package for all Scala file: *<Firstname>_<Lastname>_hw5.jar*
   If you use Scala for all tasks, please make all *.scala file into ONLY ONE *<Firstname>_<Lastname>_hw5.jar* file and strictly follow the class name mentioned above. And DO NOT include any data or unrelated libraries into your jar.
4. If you use Python, then all python scripts:
   <Firstname>_<Lastname>_task1_ TrackHashTags.py
   <Firstname>_<Lastname>_task1_ AveTweetLength.py
   <Firstname>_<Lastname>_task2.py

Grading Criteria:

1. If your programs cannot run with the commands you provide, your submission will be graded based on the result files you submit, and there will be an 80% penalty.
2. If the files generated are not sorted based on the specifications, there will be 20% penalty.
3. If your program generates more than one file, there will be 20% penalty.
4. if runtime of your program exceeds the runtime requirement, there will be 20% penalty.
5. **If you don't provide the source code, especially the Scala scripts, there will be 20% penalty.**
6. You can use your free 5-day extension.
7. There will be 20% penalty for late submission within a week and 0 grade after a week.
8. There will be 10% bonus if you use Scala for the entire assignment.