MET CS 755, Spring 2018

Assignment – 3 (20 points)

Hadoop MapReduce Programming

Due date: Mar 18, 12:00 pm

1. Description

This assignment is very similar to the last one, except that you will be using Apache Spark with Python to write and execute your codes, rather than writing MapReduce programs in Java (using Apache Hadoop).
We will again be using the data set consisting of New York City Taxi trip reports in the Year 2013.

2. Dataset

The data set itself is a simple text file. Each taxi trip report is a different line in the file. Among other things, each trip report includes the starting point, the drop-off point, corresponding timestamps, and information related to the payment. The data are reported by the time that the trip ended, i.e., upon arrive in the order of the drop-off timestamps.

The attributes present on each line of the file are, in order:

Attribute	**Description**
medallion	an md5sum of the identifier of the taxi - vehicle bound (Taxi ID)
hack_license	an md5sum of the identifier for the taxi license (driver ID)
pickup_datetime	time when the passenger(s) were picked up
dropoff_datetime	time when the passenger(s) were dropped off
trip_time_in_secs	duration of the trip
trip_distance	trip distance in miles
pickup_longitude	longitude coordinate of the pickup location
pickup_latitude	latitude coordinate of the pickup location
dropoff_longitude	longitude coordinate of the drop-off location
dropoff_latitude	latitude coordinate of the drop-off location
payment_type	the payment method -credit card or cash
fare_amount	fare amount in dollars
surcharge	surcharge in dollars
mta_tax	tax in dollars
tip_amount	tip in dollars
tolls_amount	bridge and tunnel tolls in dollars
total_amount	total paid amount in dollars

The data files are in comma separated values (CSV) format. Example lines from the file are:

07290D3599E7A0D62097A346EFCC1FB5,E7750A37CAB07D0DFF0AF7E3573AC141,2013-01-01, 00:00:00,2013-01-01 00:02:00,120,0.44,-73.956528,40.716976,-73.962440, 40.715008,CSH,3.50,0.50,0.50,0.00,0.00,4.50

22D70BF00EEB0ADC83BA8177BB861991,3FF2709163DE7036FCAA4E5A3324E4BF,2013-01-01, 00:02:00,2013-01-01 00:02:00,0,0.00,0.000000,0.000000,0.000000,0.000000, CSH,27.00,0.00,0.50,0.00,0.00,27.50

0EC22AAF491A8BD91F279350C2B010FD,778C92B26AE78A9EBDF96B49C67E4007,2013-01-01, 00:01:00,2013-01-01 00:03:00,120,0.71,-73.973145,40.752827,-73.965897 73.965897,40.760445,CSH,4.00,0.50,0.50,0.00,0.00,5.00

3. Obtaining the Dataset

We are providing two versions of the data set. The first is a small one that you can use for debugging. We strongly recommend that you write your code and execute it on your laptop using this data set first. Only once you are sure that your implementation is working on your laptop should you try the larger data set. This is going to save money, but more importantly, it is going to save time!

- **Small data set.** (93 MB compressed, uncompressed 384 MB) for implementation and testing purposes (roughly 2 million taxi trips).

This is available at Amazon S3: https://s3.amazonaws.com/metcs755/taxi-data-sorted-small.csv.bz2

or as direct S3 address, so you can use it in a MapReduce job:

s3://metcs755/taxi-data-sorted-small.csv.bz2

- **Larger data set.** (8.8 GB compressed, uncompressed 33.3 GB) for your final data analyzing (roughly 173 million taxi trips).

This is available at Amazon S3: https://s3.amazonaws.com/metcs755/taxi-data-sorted-large.csv.bz2

or as direct S3 address, so you can use it in a MapReduce job:

s3://metcs755/taxi-data-sorted-large.csv.bz2

4. Assignment Tasks:

4.1. Task 1 (5 points)

Many different taxis have had multiple drivers. Write and execute a Spark Python program that computes the top ten taxis that have had the largest number of drivers. Your output should be a set of (medallion, number of drivers) pairs.

NOTE - 1: You should consider that this is a real world data set that might include wrongly formated data lines. You should clean up the data before the main processing, a line might not include all of the fields. If a data line is not correctly formated, you should drop that line and do not consider it.

4.2. Task 2 (10 Points)

We want to know where people in NYC go on Sundays mornings between (8 am and 11:00 am) comparing to other days of the week. These places might be specific hot spots of the city like restaurant places, shopping places, concert venues, theaters, sporting venues, etc.

Your goal here is to find geographic locations ("points of interest") that people go there by Taxis. The idea is to start by dividing all of the locations into grid cells. We will compute the grid cells by a simple function to keep the Cell_id calculation as simple as possible for this task.

The following example code generates a cell ID as a string:

```
import math
def getCellID(lat, lon):
          return (str(round(lat, 2)) + "-"+str(round(lon, 2)))
```

You will need to compute the total number of drop-offs for each Cell and each time of the day/week.

Then you need to filter out the Sundays (specific time) and also the other days (same time) . Then you need to sum the result for 6 days of the week (Mon-Sat) and also for the the Sundays the same time. Finally, get the **top twenty** of them of total counts for each of the two groups (Sundays/ Not Sundays).

You will then report two sets (One for Sundays and one for other days) of your **twenty** results as a set of tuples like:

(grid_cell, total_Number_of_dropOffs, List of <point-of-interest>)

The first entry gives the boundaries of the cell, the second total number of drop offs for that day/time in the specific cell, and third final entry is a list of points-of-interest that fall in the grid cell.

To compute the closest point-of-interest, we'll use data from http://www.openstreetmap.org/

We have downloaded this data set, and added it to S3:

https://s3.amazonaws.com/metcs755/USGPSPOI.csv

or as direct S3 address, so you can use it in a Spark job:

s3://metcs755/USGPSPOI.csv

The data in this file is formatted as:

```
latitude || longitude || Name of POI || Category
```

For example:

```
40.8122222||-73.7638889||Eldridge Pool||---------park----------------
```

This data set contains GPS positions of **1,172,118 points of interest** around the United States.

Tip: At the end of your computation, you need to match the point of interests. If you can not find any point of interest for the specific cell ID you can leave it blank.

4.3. Task 3 (5 Points)

Finally, we want to know if some special events happened in the NYC City that caused unusual number of drop-offs in specific day-hour combination.

You'll compute the average drop-offs per hour-of-the-day for each grid cell, and then look for the twenty grid cell/hour-of-a-particular-day combos where the ratio of the number of drop-offs observed in that hour to the average number of drop-offs is maximized.

You will then report your twenty results as a set of tuples like:

```
(grid cell, hour, date, ratio, number, List of <point-of-interest>)
```

The first entry gives the boundaries of the cell, the second entry gives the hour of the day, and third gives the date when the extreme drop-off event was observed, the fourth gives the ratio of the number of drop-offs observed on that date to the average, the fifth gives the absolute number of drop-offs observed, and the final entry is a list of points-of-interest that fall in the grid cell.

NOTE - 3: you should use the same cell ID function as introduced in task 2.

NOTE – 4: you should clean up the data as in Task 2. When you calculate the ratio you may dived by zero. An entry inside the real-world dataset may not be something that you expect to be. Like you expect to have an int but it is there a string "NA". The following functions can be used.

```
    # Exception Handling  and removing wrong data lines

def isfloat(value):
        try:
            float(value)
            return True
        except:
            return False
```

```
def isInt(value):
        try:
            int(value)
            return True
        except:
            return False
```

4.4. Task 4 (For Advanced Student Groups)

Think about other questions that you might be able to answer by using this dataset. Describe your question (your research question) in more detail (similar to task 2), and implement it in Spark code to get answer .

5. Important Considerations

5.1. Machines to Use

One thing to be aware of is that you can choose virtually any configuration for your EMR cluster - you can choose different numbers of machines, and different configurations of those machines. And each is going to cost you differently! Pricing information is available at:

http://aws.amazon.com/elasticmapreduce/pricing/

Since this is real money, it makes sense to develop your code and run your jobs locally, on your laptop, using the small data set. Once things are working, you'll then move to Amazon EMR. We are going to ask you to run your Hadoop jobs over the "real" data using five **c3.2xlarge** machines as workers. This provides **8 cores per machine (40 cores total)** so it is quite a bit of horsepower.

As you can see on EC2 Price list , this costs around 50 cents per hour. That is not much, but **IT WILL ADD UP QUICKLY IF YOU FORGET TO SHUT OFF YOUR MACHINES**. Be very careful, and stop your machine as soon as you are done working. You can always come back and start your machine or create a new one easily when you begin your work again. Another thing to be aware of is that Amazon charges you when you move data around. To avoid such charges, do everything in the **N. Virginia** region. That's where data is, and that's where you should put your data and machines.

- You should document your code very well and as much as possible.

- You should use the Google Java Style Guide (https://google.github.io/styleguide/javaguide.html)

- You code should be compilable on a unix-based operating system like Linux or MacOS.

5.2. Academic Misconduct Regarding Programming

In a programming class like our class, there is sometimes a very fine line between "cheating" and acceptable and beneficial interaction between peers. Thus, it is very important that you fully understand what is and what is not allowed in terms of collaboration with your classmates. We want to be 100% precise, so that there can be no confusion.

The rule on collaboration and communication with your classmates is very simple: you cannot transmit or receive code from or to anyone in the class in any way---visually (by showing someone your code), electronically (by emailing, posting, or otherwise sending someone your code), verbally (by reading code to someone) or in any other way we have not yet imagined. Any other collaboration is acceptable.

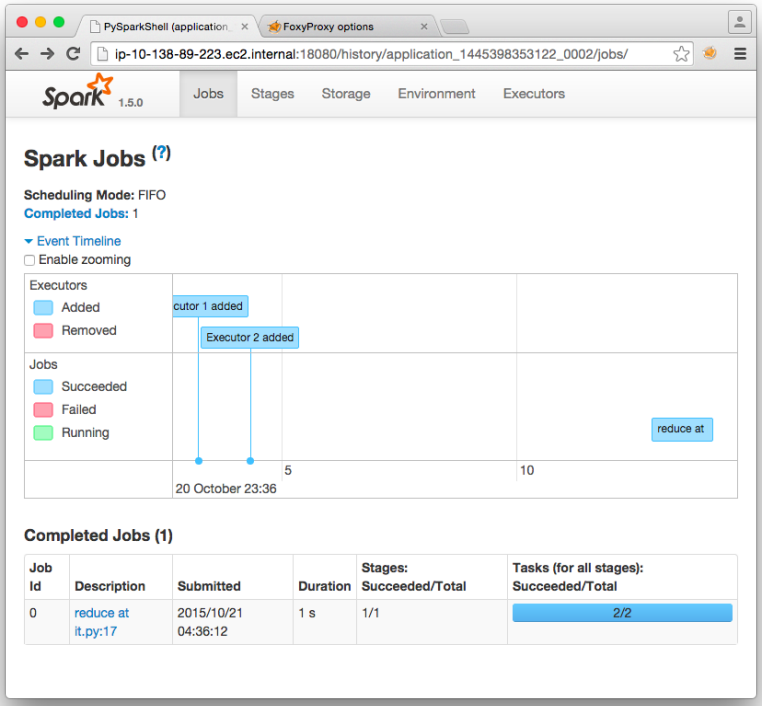
The rule on collaboration and communication with people who are not your classmates (or your TAs or instructor) is also very simple: it is not allowed in any way, period. This disallows (for example) posting any questions of any nature to programming forums such as **StackOverflow**.

As far as going to the web and using Google, we will apply the **"two line rule"**. Go to any web page you like and do any search that you like. But you cannot take more than two lines of code from an external resource and actually include it in your assignment in any form. Note that changing variable names or otherwise transforming or obfuscating code you found on the web does not render the "two line rule" inapplicable. It is still a violation to obtain more than two lines of code from an external resource and turn it in, whatever you do to those two lines after you first obtain them.

Furthermore, you should cite your sources. Add a comment to your code that includes the URL(s) that you consulted when constructing your solution. This turns out to be very helpful when you're looking at something you wrote a while ago and you need to remind yourself what you were thinking.

6. Turnin

Create a single document that has results for all three tasks. For each task, copy and paste the result that your last Spark job wrote to Amazon S3. Also for each task, for each Spark job you ran, include a screen shot of the Spark History.



Please zip up all of your code and your document (use .gz or .zip only, please!), or else attach each piece of code as well as your document to your submission individually.

6. Grading

If you get the right answer and your code is correct and you fully utilize your cluster, you get all of the points. If you don't get the right answer or your code is not correct or you don't make use of your cluster, you won't get all of the points; partial credit may be given at the discretion of the grader.

The final results should be generated by using the large dataset.