Analyzing New York City Taxi Data

Dataset

You can download the NYC taxi dataset from the link http://www.andresmh.com/nyctaxitrips/ . However, it is huge. So, we have already prepared a sample that you can find it in this link.

Each row of the file after the header represents a single taxi ride in CSV format. For each ride, we have some attributes of the cab (a hashed version of the medallion number) as well as the driver (a hashed version of the hack license, which is what licenses to drive taxis are called), some temporal information about when the trip started and ended, and the longitude/latitude coordinates for where the passenger(s) were picked up and dropped off.

We are mainly interested in each Trip's:

- Some Unique ID for the car (license)
- Pick-up location
- Pick-up time
- Drop-off location
- Drop-off time

Additionally, you will in the dataset archive you will find a .geojson file that contains the geographical boundaries of the different boroughs of New York city. This information is needed to compute the answers to the queries needed on the data.

What do we need to compute?

First, we need to compute one important metric; *utilization.* Utilization is the fraction of time that a cab is on the road and is occupied by one or more passengers. One factor that impacts utilization is the passenger's destination: a cab that drops off passengers near Union Square at midday is much more likely to find its next fare in just a minute or two, whereas a cab that drops someone off at 2 AM on Staten Island may have to drive all the way back to Manhattan before it find its next fare.

We need to compute:

1. Utilization: This is per taxi/driver. This can be computed by computing the idle time per taxi. We will elaborate on that more later.
2. The average time it takes for a taxi to find its next fare(trip) per destination borough. This is can be computed by finding the difference of time, e.g. in seconds, between the drop off a trip and the pick up of the next trip.
3. The number of trips that started and ended within the same borough,
4. The number of trips that started in one borough and ended in another one

Suggested Steps

To carry out this analysis, we need to deal with two types that data that come up all the time: temporal data, such as dates and times, and geospatial information, like points of longitude and latitude and spatial boundaries.

As the data given in the taxi ride data set is giving just the longitude and latitude of both the pickup and drop-off locations, we need to enrich this data set with boroughs of the respective locations. For this, we are going to load another data set that specifies the boundaries of each borough in GeoJSON format, data is supplied as a separate file. This data is relatively small. So, you can *broadcast* it to the different workers. It can then be loaded as a JSON object and a dataframe can be created out of it.

To enrich the taxi ride data set with pick up and drop off borough names, we need to query the GeoJSON data for the name of the borough for which the long/lat of the pick up /drop off belong. To achieve this goal, we are going to use the geometry library shapely which provides Several APIs to handle geometric shapes, among which to query about inclusion of shape in another shape.

You will need to enrich the GeoJSON record for each borough with its corresponding Shapely polygon and for each long/lat of a pickup/drop off to convert to a *Point* object in the 2D space. It is recommended to sort the GeoJSON data by polygon size in a descending order to guarantee faster access as the borough with the largest area is expected to happen more frequently in the taxi ride dataset. We can use the fact that the boroughCode property of each feature can be used as a sorting key, with the code for Manhattan equal to 1 and the code for Staten Island equal to 5. Within the features for each borough, we want the features associated with the largest polygons to come before the smaller polygons, because most trips will be to and from the "major" region of each borough. Sorting the features by the combination of the borough code and the area of each feature's geometry should do the trick.

To let Spark handle these, you need to define them as user-defined functions UDFs.

After we have put our raw data into the shape ready for analysis, we can proceed to capture our target. However, we might need a cleansing step to exclude some outliers. Given the temporal nature of our trip data, one reasonable invariant that we can expect is that the drop off time for any trip will be sometime after the pickup time. We might also expect that trips will not take more than a few hours to complete, although it's certainly possible that long trips, trips that take place during rush hour, or trips that are delayed by accidents could go on for several hours. We're not exactly sure what the cutoff should be for a trip that takes a "reasonable" amount of time. For this, we need to compute a new column "Duration" which will be the difference between pickup time and drop off time. For this, we need to use unix_timestamp function to covert the date time of the respective times into milliseconds since epoch time of Unix systems. We need that so we can treat the time as a numeric value and can do subtraction. So that, the duration will be dropoff_ts - pickup_ts. After computing, we can drop records whose duration is either negative or above a given threshold, say 4 hours, after converting them to milliseconds.

Computing utilization

Now, getting back to the computation of utilization, we need to find the idle time of each driver (taxi) in order to compute the utilization. For this, we need to sort the trips of each driver/taxi by their start time. Before that we need to shuffle the data to guarantee that all trip records for the same driver will be

available on the same worker. After we shuffle and sort the data with time, we need to compare the pickup time of record n with the drop off time of record n-1, except for the very first record. For this, we need to use the Window operator of Spark. If the difference between a ride and its subsequent ride is more than 4 hours, we ignore that as it represents a new session. Differences below 4 hours is considered though as an idle time. At the end, we need to group by destination borough and sum the idle time to address query 1.

The other three queries are much simpler and can be addressed once we enrich the data set with start/end borough names.

Grading Rubric

This project contributes 20% of the total grade. That is 20 points. The breakdown of the grade is as follows:

Item	Points
Query1 – utilization	5
Query 2	3
Query 3	2
Query 4	2
Applying optimizations like broadcasting, ordering, cleansing, etc	3
Final presentation	5
Total	20