Analysis and Visualization of NYC Taxi Trip Data.

Xianglu Kong
Department of Electrical Engineering
Columbia University
Email: xk2122@columbia.edu

Junfei Shen
Quantitative Method in the Social Sciences
Columbia University
Email: js4567@columbia.edu

Guochen Jing
Department of Electrical Engineering
Columbia University
Email: gj2249@columbia.edu

***Abstract*—New York City is mature in its public transportation, and it is an interesting task to know under what kind of situation people would most likely choose to take a taxi. In this project, we make most use of the taxi trips dataset in January, 2015: Peak hour is analyzed based on the dataset, taxi trips are visualized for every hour in a single day, and 10 most popular pickup and drop off locations are calculated using Spark clustering algorithm. We are dealing with literally 'big data' since each dataset for a single month would be more than 2GB in size, and we use HDFS and MapReduce to better organize these datasets.**

***Keywords*-big data; data visualization; clustering;**

I. Introduction

Taxis are sensors of city life.[1] As a city with possibly the busiest transportation in the world, New York City Taxi Trips data highly carries the information of where people are heading on the ground. In this project, we are going to create an intuitive picture of taxi pick up and drop off locations during 24 hours in a single day. We will be able to tell people's destinations and starting points in different time, and also the typical rush hours in NYC. Meanwhile, we implement a clustering algorithm using Spark with the data. By doing this, we seek to identify the most popular pick up and drop off places from taxi trips in NYC on January 1st, 2015. Therefore we make recommendations to taxi drivers about where might be the most active places on the coming New Year's day.

II. Related Works

There are a lot of data visualization projects on NYC OpenData(https://nycopendata.socrata.com). For example, Andrew Hill displays vehicle collisions aggregated by the time of the day using recently released NYPD motor vehicle collision data(vehicle collisions aggregated by time of day).

There are various kinds of tools for data visualization. To name some, D3.js, Google Maps, Leaflet JavaScript library, and the CartoCSS map styling language are among the most popular data visualization tools.

III. System Overview

A. Dataset

This dataset includes trip records from all trips completed in yellow and green taxis in NYC during every month from 2009 through 2014, and selected months of 2015. Records include variables covering pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, driver-reported passenger counts etc. We are using only the trips occurred with yellow cabs in January 2015.

B. City rush hour analysis

We used yellow cab's data of this January. The reason why we chose not to use data of green cabs is because green cabs are under certain regulations thus pick-up and drop-off locations do not reflect real market demand. In order to visualize demand of taxi's on New Year's day at various hours we make a bar chart showing number of pick-ups and drop-offs at each hour. To achieve this goal we use pig to process the data. First we use pig to pick out the columns that we are interested in: pick-up time, drop-off time, pick-up location, drop-off location. Then we arrange the data in chronicle order. After finishing preprocess of the data we use pig to separate the data by hour and finally we obtain number of pick-ups and drop-offs by counting number of entries in each basket. As we can see from Figure 1 and 2, the patter of pick-up and drop-off are similar which indicates the flow of traffic are bidirectional on New Year's day unlike normal work day when in morning the flow is toward downtown and outward in the afternoon. Also we can observe that the peak occurred around mid-night which makes sense because most people will be outside countdown on New Year's day. Coupled with the cluster data we acquired via $k-means$ we can unveil the most dynamic areas for New Year countdown in 2015.

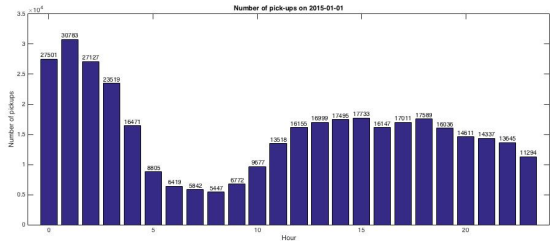


Figure 1. Number of pick-ups on 2015-01-01

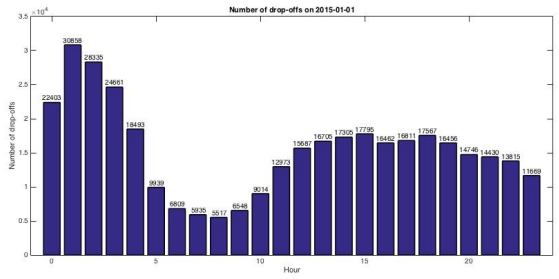
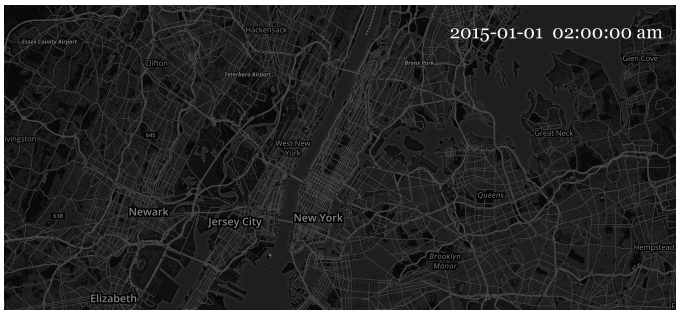
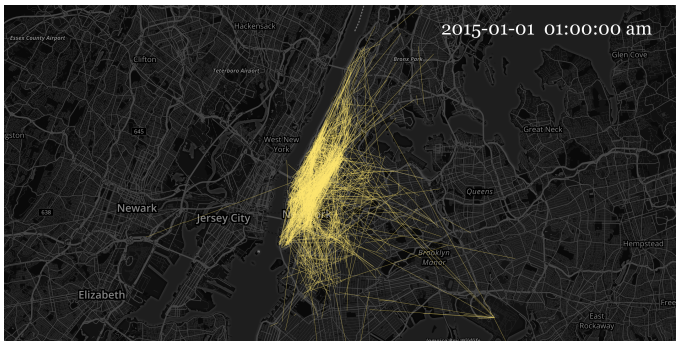


Figure 2. Number of drop-offs on 2015-01-01

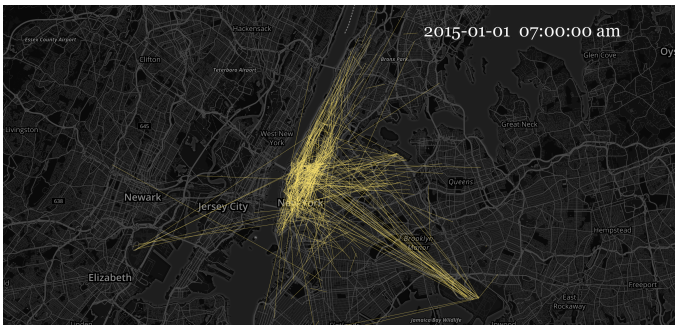
```
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:02 /countOutput/00
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:02 /countOutput/01
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:02 /countOutput/02
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:03 /countOutput/03
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:03 /countOutput/04
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:04 /countOutput/05
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:04 /countOutput/06
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:04 /countOutput/07
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:05 /countOutput/08
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:05 /countOutput/09
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:05 /countOutput/10
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:06 /countOutput/11
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:06 /countOutput/12
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:06 /countOutput/13
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:07 /countOutput/14
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:07 /countOutput/15
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:08 /countOutput/16
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:08 /countOutput/17
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:08 /countOutput/18
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:09 /countOutput/19
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:09 /countOutput/20
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:09 /countOutput/21
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:10 /countOutput/22
drwxr-xr-x   - xianglukong supergroup          0 2015-12-17 17:10 /countOutput/23
```

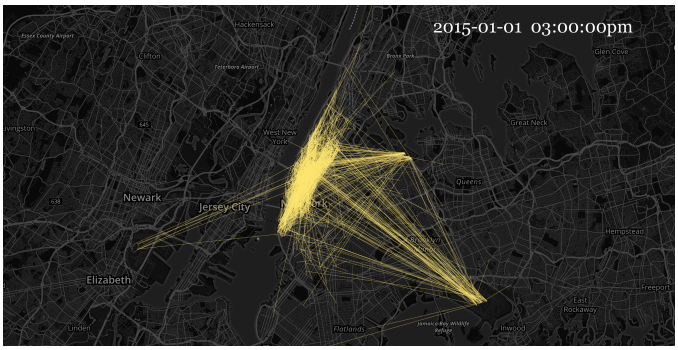
C. 24-hour Trip Visualization

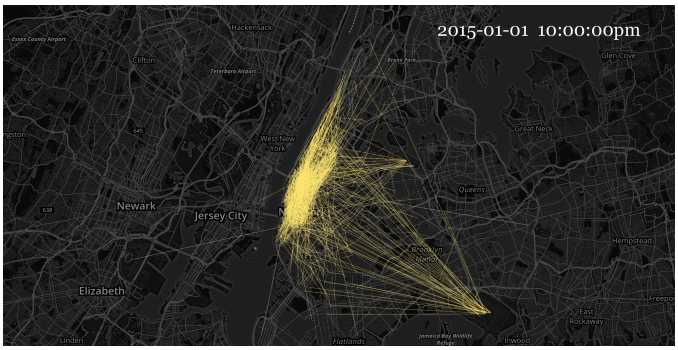




In order to visualize the trip for every hour on January 1, 2015, we first need to filter out the useful information from the whole raw dataset. We write some Pig scripts to do this selection. However, since Pig is a data control language, we cannot write a loop inside the Pig script. Therefore, we embedded the script into python and run the selection 24 times. Then we run some code in bash to collect the output data file and then convert them into one single JavaScript file, which can be used as the source data in the javascript code for drawing maps. Some snippets of the code are listed below.








```
# getLocInfo.py
from org.apache.pig.scripting import *


P = Pig.compile("""
tripdata = LOAD '/input/yellow_tripdata_2015-01.csv' USING PigStorage(',') AS (
        VendorID,tpep_pickup_datetime,tpep_dropoff_datetime,passenger_count,
        trip_distance,pickup_longitude,pickup_latitude,RateCodeID,
        store_and_fwd_flag,dropoff_longitude,dropoff_latitude,payment_type,
        fare_amount,extra,mta_tax,tip_amount,tolls_amount,improvement_surcharge,
        total_amount);

dayRecord = FILTER tripdata BY (tpep_pickup_datetime matches '$time')
AND (pickup_longitude!=0)
AND (pickup_latitude!=0)
AND (dropoff_longitude!=0)
AND (dropoff_latitude!=0)
AND (pickup_longitude!=dropoff_longitude OR pickup_latitude!=dropoff_latitude);
```

```
orderByTime = ORDER dayRecord BY tpep_pickup_datetime;
posInfo = FOREACH orderByTime GENERATE tpep_pickup_datetime,tpep_dropoff_datetime
      ,pickup_latitude,pickup_longitude,dropoff_latitude,dropoff_longitude;
STORE posInfo INTO '$outpath';
""")

params = {}
for i in range(23):
    params["time"] = "2015-01-01 " + ("%02d" % i) + ":*"
    params["outpath"] = "/output/dayRecord/" + ("%02d" % i)
    print(params["time"])
    print(params["outpath"])
    bound = P.bind(params)
    stats = bound.runSingle()
    if not stats.isSuccessful():
        raise 'failed'


#!/bin/bash
set -e
for i in $(seq -f "%02g" 0 23)
do
    ./bin/hdfs dfs -get /dropOffCountOutput/$i/part-r-00000 /Users/xianglukong/
          Documents/BigData/FinalProject/d3js_project/js/dropOffCountOutputs/
          hour$i
done

cd ~/Documents/BigData/FinalProject/d3js_project/js/
for i in $(seq -f "%02g" 0 23)
do
    awk 'NR % 12 == 0' rawOutputs/part-r-000$i > "pathxxinput/path"$i"input"
done

python parse.py
```

We use Mapbox.js to draw the maps, which provides nicely customized maps. Based on these beautiful maps as canvas, we create our own layers of data polylines, and call a function iteratively to draw the each trip as a polyline with a time offset of 1ms. In each loop we delete the previous layer of lines and add a new layer on top of the map.

```
function next() {
    // jQuery TO UPDATE THE TIMESTAMP

    map.removeLayer(polylines);
    polylineArray = [];
    pairs = paths[hour];
    for (var i = 0; i < pairs.length; i++) {
        var generator = new arc.GreatCircle(
                obj(pairs[i][0]),
                obj(pairs[i][1])
                );
        var line = generator.Arc(100, { offset: 10 });
        var newLine = L.polyline(line.geometries[0].coords.map(function(c) {
            return c.reverse();
        }), {
            color: '#ffe766',
            weight: 0.5,
            opacity: 0.7
        });

        polylineArray.push(newLine);
    }
    polylines = L.layerGroup(polylineArray);
    polylines.addTo(map);

    for (var i = 0; i < polylineArray.length; i++) {
        var newLine = polylineArray[i];
        var totalLength = newLine._path.getTotalLength();
        newLine._path.classList.add('path-start');
        newLine._path.style.strokeDashoffset = totalLength;
        newLine._path.style.strokeDasharray = totalLength;
        setTimeout((function(path) {
            return function() {
                path.style.strokeDashoffset = 0;
            };
        })(newLine._path), i * 1);
    }
        if (++hour >= paths.length) hour = 0;


}
```

From the output of trip visualization, we can see that the worst traffic condition happens in early afternoon, while the traffic condition appears to be the best between 2 and 3 in the morning. But even at that early time, there are still a lot of taxi trips on the Manhattan Island.

On the other hand, traffic conditions on manhattan island are always very crowded, and we can not see very clearly which part of the city is more taxi-demanding. It is quite obvious that three airports stand out from all the other locations within Manhattan though, which are JFK Airport, La Guardia Airport and Newark Airport. It seems that New York people are quite inclined to choose taxis when they travel from home or office to airports.

D. Clustering Popular Locations

1) Clustering in General: Clusteringindicates a statistical method for finding subgroups in a larger group of data to achieve the goal that given a certain measure of similarity, the data points in the same subgroup i.e. cluster are more similar to each other, while data points in different subgroups share less similarities. It is a common technique in exploratory data analysis, as well as an unsupervised machine learning method.

Rather than a specific algorithm, clustering is more of a mindset of dealing with data. There are various algorithms implementing the clustering idea, differing greatly in similarity measures and definition of cluster. And the significant variety of defining "distance" and "cluster" results in great diversity of clustering algorithms. To name some: connectivity models, which is based on distance connectivity; centroid models, where clusters are represented by individual mean vectors; distribution models, soft clustering methods in which clusters are statistical distributions; density models, clusters of whom are connected dense regions in data space; graph-based models, where distance measures are derived from topological values of the nodes in graph; etc.

Appropriate clustering model should be chosen depending on the specific problem to be solved. And the clustering analysis is not totally automatic: parameters, such as number of clusters, need to be modified on demand to allow the model to achieve the best results.

2) K-means: Given a bunch of data points, where each data point is a multi-dimensional numerical vector, k-means clustering solves the problem of finding a best partition of the data points into k groups, minimizing the cost of the model. Cost of k-means model is usually defined as the sum of the distance of each data point from the centroid it is assigned to.

Mathematically, the goal k-means clustering tries to achieve is

$$\arg\min_{S} \sum_{i=1}^{K} \sum_{x \in S_i} \|x - m_i\|^2$$

where S is the set of all data points, and m_i is the centroid of points assigned to cluster . Given the k initialized mean vectors, the k-means algorithm works in a "coordinate ascendant" way. There are two steps in each iteration of k-means algorithm: first assign data points to clusters, and second update the centroids of clusters.

In the first step, after calculating the distance of each data point - centroid pairs, data points are assigned to a cluster that has the shortest distance (usually Euclidean distance)

from the data point. Intuitively, given the fixed centroids, data points are assigned to the nearest centroids in this step.

Mathematically, this step is

$$S_i^{(t)} = \{x_p : \left\|x_p - m_i^t\right\|^2 \leq \left\|x_p - m_j^t\right\|^2 \forall_j, 1 \leq j \leq k\},$$

where each data point x_p is assigned to a cluster $S^{(t)}$ represented by a centroid. In the case that two clusters yield the same shortest distance, one centroid is selected at random.

In the second step, centroids of clusters are updated according to data points previously assigned to them. The new mean vector for a cluster is calculated by taking the arithmetic mean of all data points assigned to it. Due to the least-square nature of arithmetic mean, cost of the model given the data point assignments, represented by within-cluster sum of squares, is also minimized in this step.

The mathematical formula for step two is

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

When the assignments of data points to centroids do not change any more, the algorithm has come to a convergence. For any initialization, the algorithm must converge after finite iterations, because the cost of model is minimized in both step 1 (keep centroids fixed) and step 2 (keep data points assignment fixed). Finally the algorithm will converge to a local optimum.

However, there is no guarantee that global optimum can be found in a single run of k-means. Final results of the clustering may vary, depending on how the clusters are initialized. So given the high efficiency, the algorithms is usually to be run several times with the same set of data points but under different initializations for a better result. In the worst case, however, the convergence can be slow, which is rare. Fortunately, in general, the k-means algorithm has polynomial running time.

As for initialization methods, there are two commonly used ones: Forgy and Random Partition. Forgy randomly chooses k data points from the dataset as initialized means, while Random Partition assign random clusters to data points and go on to update cluster centroids. For standard k-means algorithm, Forgy is preferable, which indicates that the algorithm starts off from randomly selected data points as initial cluster means.

Also k-means clustering is an example of generalized EM algorithm. The "assignment" step can be thought as "expectation", and the "update" step is "maximization".

In addition, using distance measures other than Euclidean distance or squared Euclidean distance may cause the algorithm's failure in converging.

3) K-means in combination of NYC Taxi Trips Dataset using Spark: In our NYC Taxi Trips Dataset, previous visualization has shown that taxi trips in Manhattan on New Year's Day are too busy and dense to figure out popular locations on the island, except two outliers: the JFK Airport and the LGA Airport. If more concrete locations are to be identified, extracting fewer important points from the massive data points is desirable, where k-means clustering can do a good job.

K-means clustering s conducted to find popular locations, based on pick up and drop off locations' longitude and latitude as data points' features.

Since the csv data file is as large as 2 GB, the clustering is done with Spark.

First the csv file is loaded as a text file, with header line removed, date time variables converted into Python datetime objects and longitude and latitude converted from original string variables into float variables.

Second, only pick up longitude, pick up latitude, drop off longitude and drop off latitude are selected into the data frame, while all other variables like tip amount, number of passengers etc. are ignored. Also only the taxi trips occurred between 9 in the morning and noon on January 1 are selected by filtering the date time variable. Here the map() method of Spark rdd object are used twice for the transformation.

Finally the k-means clustering is done in Spark with a single line of code. The value of k is set to 10 for no specific reason. The k-means clustering is set to random initialization and run for 4 times. The output contains the longitude and latitude of final cluster centers.

4) Clustering results: The longitude and latitude values along do not make intuitive sense, so visualization is made with these data. From a map of NYC with these cluster centers plotted, it is clear that apart from JFK Airport and LGA Airport, pick up centers also include Wall Street, East Village, Flat Iron Building, Times Square, Grand Central Terminal, Upper East, Upper West and Harlem, and drop off cluster centers include Jersey City, Brooklyn, SOHO, Penn Station, Grand Central Terminal, Upper East, Upper West and Bronx.

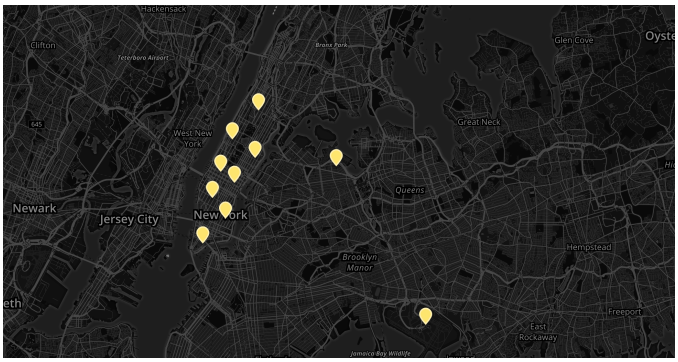


Figure 3. Clustering of taxi pickup

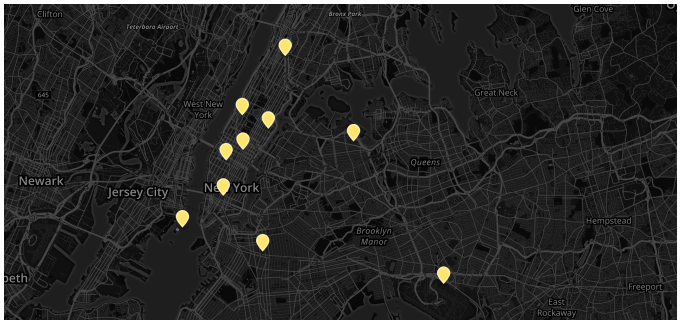


Figure 4. Clustering of taxi dropoff

IV. Software and language

There are many ways to do the data visualization. In our project we use mapbox, which is a platform that provides customized online maps. It is widely used by many companies in the industry, such as Foursquare, Pinterest and Uber technology. It is based on Leaflet.js, which is an open source javascript library, similar to google map, used to build different web mapping applications.

In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's multi-stage in-memory primitives provides performance up to 100 times faster for certain applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well-suited for machine learning algorithms.

Acknowledgment

We would like thank professor Lin for the great lectures in the past semester, as well as instructions and directions on Hadoop ecosystem, data visualization and other big dat tools. We are also thankful to the TAs for giving suggestions when we ran into trouble. Furthermore, we are glad that there are so many great works on the internet, which brings us a lot of inspiration.

Appendix

$$https://youtu.be/HUNbyokYlXM$$

References

[1] Visual Exploration of Big Spatio-Temporal Urban Data: A Study of New York City Taxi Trips