CS123 Project: Analyzing Chicago Taxi Data

Salman Arif, Andrew Chuang, Francesco Scivittaro

Link to GitHub Repository:
https://github.com/salman-arif/CS123-Project
Description of Chicago Taxi Dataset:
http://digital.cityofchicago.org/index.php/chicago-taxi-data-released/
Link to Dataset:
https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew/data

Description of Dataset
The dataset we chose for our CS123 project was the Chicago Taxi Data on trips between 2013-2017. This 40GB dataset contained 109,709,078 rows representing a single ride with 23 columns each representing a single variable of the ride. Notable columns included a unique taxi ID for the taxi, a unique trip ID for each trip, trip distance, trip length in seconds, community area pickup/drop-off, and coordinates of the pickup and dropoff points. We were primarily concerned with the pickup and dropoff coordinates for our first project hypothesis and the unique taxi ID and community area pickup/dropoff points for our second project hypothesis.

Basic Summary Statistics:

- Month with the lowest average fare: December ($13.84)
- Average tip: $12.46
- Most common payment types: Cash followed by Credit Card
- Average Ride Length: 745 seconds
- Most Common Pickup/Dropoff Points: Loop and Near North Side neighborhoods

Hypotheses Tested
Hypothesis #1: What neighborhoods in Chicago are the most likely to be congested with traffic? Can the most popular routes be tracked by the intersection of all possible trips taken by Chicago taxis?

Hypothesis #2: Supply Demand Hypothesis
Where in Chicago are drivers least likely to find passengers? Are there any misconceptions that drivers have about certain neighborhoods that cause them to miss potential passengers? Can we analyze this problem by determining the supply and demand of taxi rides in the area?

BigData Approaches Used
For our first hypothesis, we used MapReduce (MRJob). We compared all pairs of rides and plotted their intersection points into a heat map over a map of Chicago. We also utilized subsampling, which allowed us to test our code on more manageable samples of data before running the code on Google Cloud Dataproc.

Challenges Faced

For the first hypothesis, we faced a number of conceptual challenges. Comparing all possible pairs of trips was computationally intensive and required the use of MapReduce. Initially, we generated a separate csv file with two columns using a nested pair of “for” loops. In order to utilize this file, we wrote a MRJob file that took in this CSV file and pulled out the two columns of interest from a row in the file. This approach failed to process a file with only 500 lines within 1 hour, which meant that it would not work well for the full dataset. We decided to use the fact that MRJob took files line by line to loop through the file for every row rather than utilizing an additional intermediary file. The approach allowed us to successfully process the 2k, 5k, and 50k samples within 1 hour. During our presentation slot on 6/2, we had the opportunity to talk with two other groups who had faced similar problems comparing all possible pairs of records. One group (Fiona Jack and Peter) suggested labeling all rows in order to keep track of the pairs (trip intersections are commutative) and avoid double processing pairs.

A major challenge we faced was that of how best to represent the area around an intersection. In order to make our results as accurate as possible, we wanted to use the angle of intersection between two routes to estimate how similar two routes were likely to be. In other words, for two trips that intersected at very narrow angles, we assumed that the two cars likely used the same street or highway for much of the trip, and that the zone this street or highway was in was likely to be in a rectangle surrounding the area where the trip intersected. However, because these polygons were often slanted or somewhat irregularly shaped, passing their coordinates to mapreduce was not trivial. Given a polygon, the minimum and maximum extent of each polygon both in the north-south and east-west directions was calculated and then a nested loop over the new, sometimes much bigger, polygon was performed in increments of one-hundredths of a degree of longitude and latitude. For each coordinate in this nested loop, we had to check if it was actually in the original polygon, and then if it was, we passed it to the combiner and reducer. This meant that for every pair of intersecting trips, we had to perform a nested loop and ended up discarding the results of many of the individual iterations. For this reason, any single pair of trips took a long time to compute, and when this fact is combined with the quadratic scaling of the algorithm, we had to deal with long runtimes. A 100k sample of taxi trips took approximately 8 hours to process using Dataproc with 10 nodes, and doing the full 109 million rows would have taken several orders of magnitude longer. Using Dataproc regardless performed far better than simply running our algorithm on a VM. A 5,000 line sample took 25 minutes to finish running on a VM, and a 50,000 line sample would have taken over 41 hours to complete on a VM.

We also faced a problem with the best package to use to generate the heatmap. Before our presentation, we found that many of the heat map packages that we had been considering required a loaded list of points, rather than a list of points with “weights” associated with each point. Due to the high number of points we were covering in our heatmap, loading an list of over one hundred million repeating points would be very unrealistic. Instead, we used MapReduce to sum up the counts over each unique coordinate on the heatmap grid in order to have a short list of points with weights associated to each coordinate. The Seaborn and Gmaps libraries allowed

us to plot coordinates and weights, so that we could load a very short CSV file into memory to make the heatmap, instead of having to load a file millions of lines long into memory.

The biggest assumption made with regards to this hypothesis would be the accuracy of representing taxi rides through straight lines between the pickup and dropoff points. Although Chicago's downtown roads follow a straight line grid system, Interstate 90 and 94 are not straight and cut across many side roads in a curve that we could not represent accurately. Further steps for our project could involve modeling a more precise representation of the taxi trips in order to better represent possible spots for congestion in the Chicagoland area.

Another assumption made for the first hypothesis is that taxi trips can at least somewhat accurately represent the types of trips non-taxi cars are likely to take. This was an important assumption because part of the goal of the first hypothesis was to identify where traffic congestion was likely to occur. However, because taxis likely go to airports, hotels, tourist destinations, entertainment districts of downtown much more often than cars driven by Chicago residents would, our model is likely to overestimate how much congestion surrounds these places in non-tourist seasons and on weekdays, and would in turn underestimate congestion in locations that must be traversed by rush hour traffic. While this can be considered a very serious limitation in our methodology, and in particular this assumption limits the scope of conclusions we can draw, our results do show that congestion is likely around and along major highways, which at least intuitively indicates taxi data is a "good enough" proxy for commuter or "civilian" traffic.

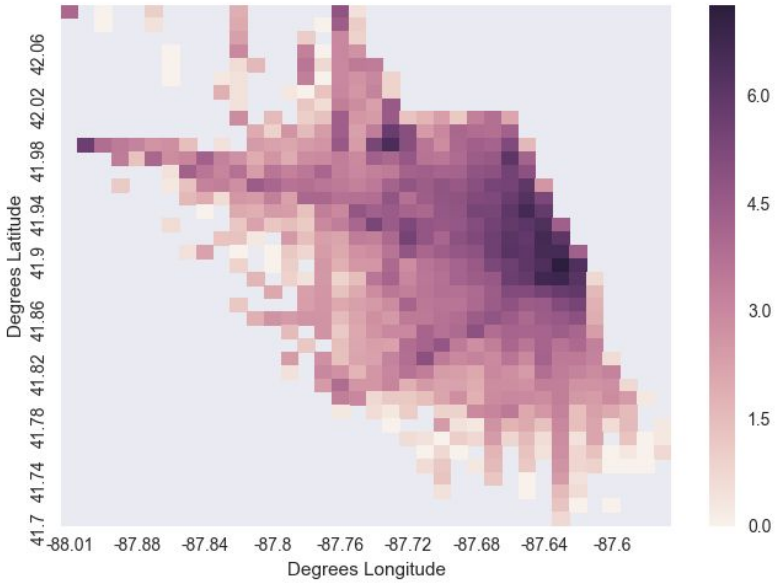
For the second hypothesis, we needed to make a number of assumptions in order to make our approach tractable. First, we assumed the individual taxi ID represented an individual driver. In reality, taxi companies own taxi medallions which then allowed them to buy a taxi and lease it out to individual drivers in shorter eight hours shifts. Second, we assumed that individual taxi drivers did not take a break during their entire shift. Our analysis was not focused on the number of breaks that drivers took and we instead assumed that they would spend their shifts attempting to maximize the amount of paying trips they could make.

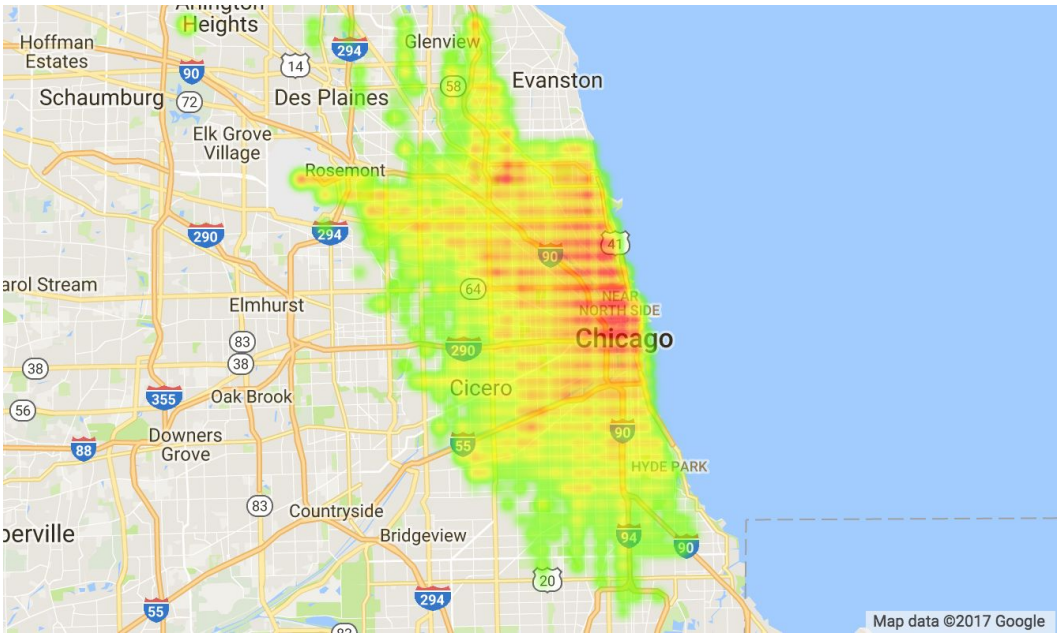
Results

Hypothesis #1:

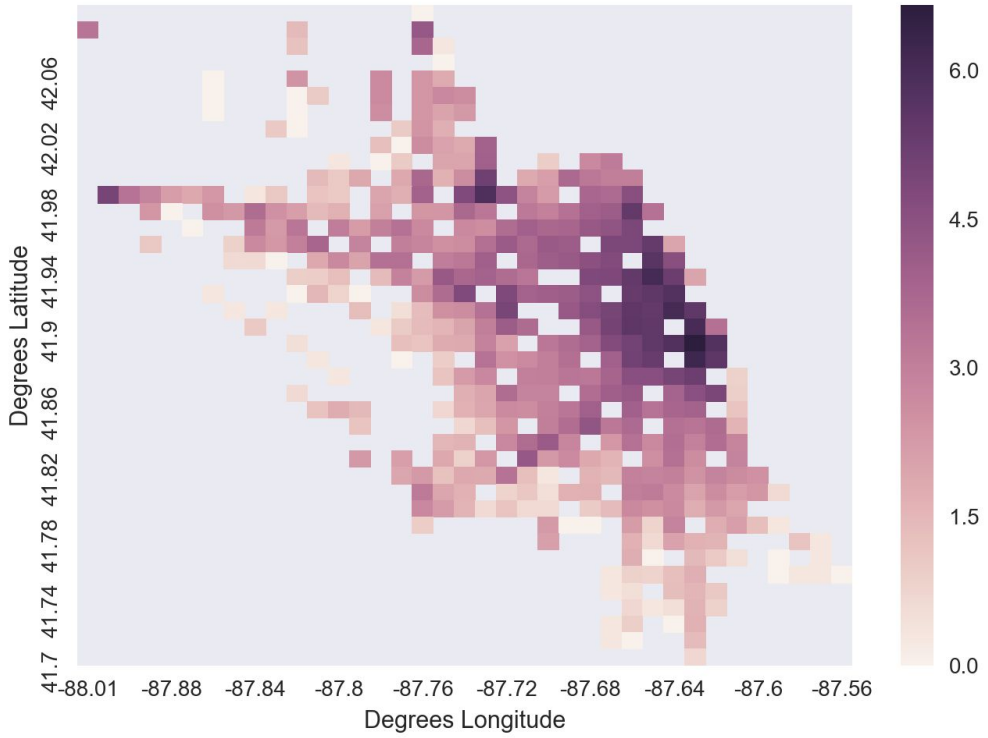
Before our presentation, we were able to make the two following heatmaps only using the Seaborn package to represent the intersections we found between rides. We later also made maps using Gmaps.

Heatmaps created using Seaborn and Gmaps packages
(Log-transformed densities)
100k Subsample - 8 hour runtime, 10 nodes





50k subsample - 2 hour runtime, 10 nodes



The results from these initial heatmaps were not too surprising; high traffic areas like the loop and the area surrounding O'Hare Airport were more densely represented on both heatmaps. The darkest points on the heatmap are around the Lincoln Park, Near North Side and Wicker Park neighborhoods. The route from O'Hare to downtown Chicago is also well represented in this heatmap. The results when overlaid onto the map of Chicago also seem to follow the major highways through Chicago pretty closely. This would seem to indicate that the highways are placed fairly optimally. If we had seen a very dark branch in a location far from any highways, that would likely indicate an opportunity for improving congestion problems in Chicago.

One unexpected point of interest was a dark square at around the coordinates (41.97, -87.75). After comparing this spot on the heatmap to an actual map of the roads and highways of Chicago, we found that this spot represented the intersection between Interstate 90 and 94. A point of intersection far from the neighborhoods with the most pickups suggests that the algorithm is capable of finding locations that stand between many popular pickup spots and destinations. While it could be a coincidence that the two interstates meet in this location, it is also possible that the two interstates are in this location precisely because it is a spot that traffic flowing both East-West and North-South must pass through. However, our preliminary results suggested that the primary point of congestion for most rides surrounds the popular pickup and dropoff points.

The algorithm used to construct the heatmaps scaled approximately quadratically. A 50,000 subsample of the data was processed on 10 nodes in approximately 2 hours, and a 100,000

subsample was processed on 10 nodes in about 8 hours. Processing the full dataset would most likely take an unreasonably long amount of time, even using 10 powerful nodes in the dataproc. The nodes we used were ‘n1-standard-4’ type, which meant they had more computational power than the most basic nodes.

Hypothesis #2:
Our approach to the supply and demand of taxi passengers in different neighborhoods started by finding the likelihood of driver finding a ride in the same neighborhood within 30 minutes of their last ride. The results from this analysis were not surprising; the Loop neighborhood (32), Near South Loop neighborhood (33) and Near North Side (08) neighborhoods were the best spots for taxi drivers to find rides within thirty minutes of dropping off their last ride. The Near West side neighborhoods were also good for drivers with a high probability of pickup.

It is important to note that due to sampling, these numbers are systematically biased downwards. While the full dataset was not processed in time with this algorithm, we would expect these probabilities to be higher when the full data is used.

Table of Results from 30 Min Likelihood on 5M Subsample		
	Community Area Name	Number of pickups in community area within 30 minutes divided by total number of pickups
28	Near West Side	0.1899
32	Near South Side	0.1771
33	Loop	0.1617
6	Lake View	0.1689
8	Near North Side	0.1921