Chicago Taxi Trips Report

Edwin Gavis, Mike Gu, Maya Jones, Madeline Kim

Data and Hypothesis

Dataset:

We used the City of Chicago's Taxi Trips dataset. The dataset as a whole contains 113 million rows and 23 columns, amounting to 46.3 GB. However, for our calculations we focused on the most recent two years of data, 2016 and 2017, for a combined 27.5 million rows and 11.8 gb. Columns which were important in our project were: trip ID, trip start timestamp, trip seconds, as well as pickup and dropoff latitude and longitude.

Hypothesis:

Our goal was to create a program that could predict areas of high traffic in Chicago by examining the intersection of different taxi trips. We predicted that areas of traffic will be most heavily congested around airport routes, such as I-294 O'Hare Way to Irving Park Road, and along major roads, such as Michigan Avenue and Milwaukee Avenue.

This hypothesis was tested in two ways. First, the program identified the average length of a taxi trip from one area to another, by time of day. Then, trips that were abnormally long were compared to the average in order to identify intersections between the trips and predict areas of high congestion. Second, the program identified a random sample of trips from the larger dataset (unfiltered) and found the intersection of trips within this sample, looking simply at frequency of intersection locations alone. By testing our hypothesis in both these ways, we hoped to create a more robust predictor of Chicago traffic and see if we could identify a difference between predicted congestion (from the random sample) and demonstrated congestion (from trips that took longer than average).

Analysis Process

Determining Average Taxi Trip Length

In order to determine the average trip length from one geographic area to another, we first had to segment the map into equally sized bins so that trips were comparable. In order to do this, we utilized geohashing. Geohashing is a public domain geocoding system encodes coordinates letter-number strings. Using the geohash2 library, one can assign an arbitrary level of precision, and nearby coordinates will be hashed to the same letter-number string. For this project, we used bins of size 153m by 153m.

Additionally, we had to segment the trips by time of day, in order to ensure we were comparing like trips. To do this, we used WBEZ's analysis of traffic levels over the course of a day to place trips into three bins: morning (5:00 - 11:59), afternoon/evening (12:00 - 21:59), and night (22:00 - 23:59, 00:00-4:59).

In order to calculate averages, we used MapReduce via the MRJob library in Python. After creating a key that consisted of a trip's start geohash, end geohash and the time of day at which it occurred which we called 'final geohash', our mapper and combiner tracked each key's frequency and the sum of total seconds for each type of trip, before calculating the averages in the reducer.

Determining Standard Deviation

Created an averages dictionary from an output file of average trips lengths, using final geohash as key, and the average trip time for that final geohash as value. Then using mrjob MapReduce on the dataset, we subtracted each trip time from the average time using the averages dictionary for the final geohash for that trip, and squared the resulting value to compute the variance for each trip. After summing the variance for each final geohash, we divided by the number of trips for that geohash and took the square root to obtain the standard deviation for each final geohash and created an output file with this information.

Filtering Trips

Created a standard dictionary from an output file of standard deviation, using final geohash as key, and the standard deviation for that final geohash as value. Created same averages dictionary as standard deviation step. Using mrjob MapReduce on the dataset, we subtracted each trip time from the average time for each final geohash. After summing the difference from the average trip time for each final geohash, we returned the trips that had trip times greater than or equal to the standard deviation for the corresponding final geohash from the standard deviation dictionary and created an output file with the filtered trips.

Identifying Intersections

Using csv file output from the filtered trips step, we calculated the intersection points (coordinates) between every qualifying pair of trips. For each trip we extracted the origin and destination coordinates and generated a linear equation in the format (ax + by = c). From a duplicate of the filtered csv we calculated the intersection (if possible) of the given trip with each trip in the second csv file, from which extracted the origin and destination coordinates and calculated a second linear equation, also in the format of ax + by = c. Provided that the trip IDs from the first and second csv files were not identical

and the two lines were not parallel or the same, a matrix system was created from these linear equations and the intersection solved using Cramer's rule. From our resulting intersections, we filtered out those coordinates falling outside a rectangle (-87.666025<longitude< -87.522251; 41.639517<latitude< 41.972081) roughly encompassing the entirety of the city of Chicago and divided the counts by 2, to account for duplicate calculations, producing a list of key value pairs of "longitude, latitude" and frequency count.

Cloud Computing

Algorithms

Processed parallelizable tasks across dataset using a cluster. Resource pooling. Multitenancy. Device and location independence. Geohash - hierarchical spatial data structure which subdivides space into buckets of grid shape.

Big Data Approaches

For our big data approaches we used MapReduce via Yelp's MRJob library versions 0.5.1 and 0.6.2 for Python 3.6. We then ran our programs on a Google Cloud DataProc cluster using 10 to 15 nodes, each with 15 or more gb of RAM. The data and output files were stored in Google Cloud buckets and a 200gb persistent disk. MRJob bootstrapping was used to install Python 3.6, the MRJob, mr3px and geohash2 libraries and other dependencies on each node upon initialization.

Challenges

Although we were able to calculate the averages, standard deviations and filter the 27.5 million rows on the Google DataProc cluster, calculating all the possible intersections between trips proved much more computationally expensive. We therefore randomly sampled 50,000 trips from both the filtered (2.3m slow trips) and non-filtered (28m trips) datasets and then compared each row in the sample to each other, resulting in 2.5 billion trip comparisons or a bit over an hour's run time on a Google DataProc cluster with 10 nodes. Since those ten nodes, each significantly more powerful than our virtual machines, took over an hour to complete the analysis in tandem, it seems rather unlikely that we would have been able to complete this analysis on a single machine in a reasonable timeframe.

Results

The 5 billion total trip comparisons identified over 19 million unique intersections inside the geographic coordinates we'd defined as Chicago. These were then compared via GIS visualizations using the commercial data visualization software Tableau 10.4.6. The following visualizations confirmed what we and likely most Chicagoans had long suspected, that the absolute worst traffic is found in the Loop and on the avenues entering the Near North Side (e.g. Milwaukee, Clybourn), while the roads to and from O'Hare International Airport contribute significantly to congestion. Midway International Airport's contribution was less visible in our analyses, however, and although the South Side had significantly fewer trips in both our samples, most of its congestion followed Lake Shore Drive.

However, all these findings are subject to flaws in the dataset and the imprecision of our methods, including our heavily abstracted method for predicting trip intersections, which does not account for variations in often nonlinear routes. Some trips in the dataset did not include origin and/or destination coordinates, and these trips had to be excluded. In addition, these trips were self-reported. While the City of Chicago believes that most trips have been included, some bias may have been introduced through this data collection system. Lastly, while we did sample randomly from both the significance-filtered and unfiltered sets of trips when calculating intersections, it would have been ideal, given more computing resources and time, to have been able to run the analyses on all the taxi trips from 2016 to 2017.

Visualizations

Figure 1

Figure 1: Marks on the map show intersections of 25,000 or more trips, in either sample. Size of circle corresponds to log(number of intersections). Figure 1a shows geographic location of trips. For each dot, color encodes number of trips significantly slower than the average for those particular coordinates. More red means most trips at that location were slower than the average; blue means the majority of trips intersecting at that point were not significantly slower relative to the respective averages for their origin/destination and time of day. Figure 1b shows the same map as Figure 1a, but with a rough overlay of streets drawn from Google Maps. Intersections were visualized in Tableau 10.4.6.

Figure 2

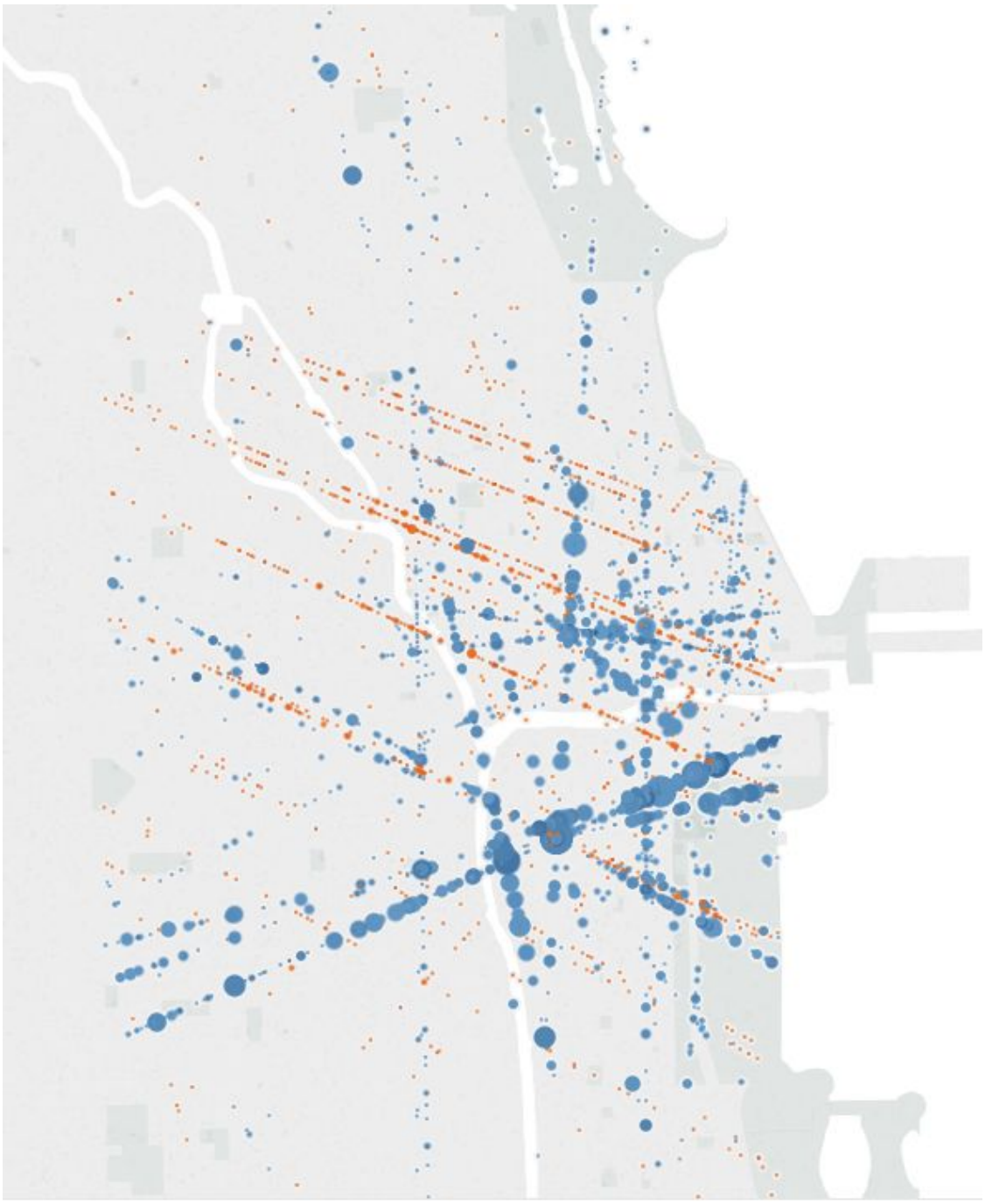

Figure 2: Same specifications as figure 1, except now zoomed in on the loop and including all intersections that appear in both samples (4.4 million) and are the meeting point for 2,500 or more trips. The orange diagonal lines may correspond roughly to some of Chicago's major diagonal avenues (e.g. Milwaukee, Clybourn, Lincoln, Clark). Intersections visualized in Tableau 10.4.6.

References

Bentley, C. (2015, January). *When is Chicago-area Traffic the Worst?* Retrieved from: https://www.wbez.org/shows/curious-city/when-is-chicagoarea-traffic-the-worst/73cf66f9-8a44-4227-8619-68220de78ad3

City of Chicago Data Portal. *Taxi Trips*. Accessed: May 20, 2018. Retrieved from: https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew/data