Ride Hailing Supply and Demand Forecasting using Didi-Tech Dataset
Data Collection Report

Kimberly Williamson, Gopal Menon
Data Mining
Spring 2017, University of Utah

I. Background

Didi Chuxing is the leading ride hailing company in China and processes over 11 million trips, plans over 9 billion routes and collects over 50TB of data per day. They organized a worldwide algorithm challenge in the year 2016 [1] for forecasting ride supply and demand.

II. Project Scope

The project team will use the 2016 Didi algorithm competition dataset to forecast taxi trip supply, demand, and expected fare for any given date, time, and location using the regression methods that will be covered in the Data Mining 2017 Spring semester at the School of Computing of the University of Utah. The accuracy of the forecast will be evaluated using the same average forecast error metric that was used in the competition.

III. How the Data was obtained

Training and Testing data was provided by Didi as part of the 2016 competition.

IV. Data Size

TABLE I
Train and Test number of rows

Dataset	Orders	Traffic	Weather	Cluster Map	POI
Train	8,540,614	193,553	4811	66	66
Test	557,985	8381	78	66	66

V. Data Format

Didi divides a city into n non-overlapping square districts $D = \{d_1, d_2, \ldots, d_n\}$ and divides one day uniformly into 144 time slots $t_1, t_2, \ldots, t_{144}$, each 10 minutes long. The training set contains 3 consecutive weeks of data for City M in 2016, and we need to forecast the supply-demand gap for a certain period in the 4^{th} and 5^{th} weeks of City M. Following are the tables in the dataset in tab separated format [1]:

1) **Order Info:** There are n orders in the dataset, such that the $O = \{o_1, o_2, \ldots, o_n\}$. O is a container where o_n is a container with attributes representing Order Id, Driver Id, Passenger Id, Start District Id, Destination District Id, Price, and Time.
2) **District Info:** This table will be stored as an associative array, where the District Id(value) is mapped to the District Hash(key). $DI = \{di_1, di_2, \ldots, di_n\}$ and $di_n = \{$District Hash:District Id$\}$.
3) **POI Info:** The points of interest(POI) for each district. The POI will be stored as a container, where each district,d, has its own POI associative arrays. $POI = \{poi_1, poi_2, \ldots, poi_d\}$ where $d \in D$. $poi_d = (pc_1, pc_2, \ldots, pc_n)$, where there are n POI classes and $pc_n = \{class_n : \#$of facilities$\}$.
4) **Traffic Jam Info:** The traffic data will be stored as a container, where each d, t combination has its own traffic jam of associative arrays. $TJ = \{tj_{1,1}, tj_{1,2}, \ldots, tj_{d,t}\}$ where $d \in D$ and $t \in T$. $tj_{d,t} = (tjl_1, tjl_2, \ldots, tjl_n)$ where there are n traffic jam levels and $tjl_n = \{level_n : \#$of congested roads$\}$.
5) **Weather Info:** The weather will be stored as a container, where each time slot,t, has its own weather list. $W = \{w_1, w_2, \ldots, w_t\}$ where $t \in T$. $w_t = (temp, pol, wt_1, wt_2, \ldots, wt_n)$ where $wt_n = \begin{cases} 1, \text{if the weather type of n is in the timeslot} \\ 0, \text{if the weather type of n is not in the timeslot} \end{cases}$

VI. Data Processing

Our plan is to create a single OrderItem,oi, for each order in O. The other containers will be joined on either d or t or both in the case of traffic data.

VII. Data Simulation

The dataset has missing or incomplete data due to orders that were not fulfilled by drivers, multiple orders generated by a customer for the same ride, orders generated by third party applications, and data that could not be collected due to glitches or technological limitations. Unfullfilled orders will be removed and not used as inputs into the regression models. We plan to use matrix completion techniques to fill in missing data when data is required. The Didi dataset is already separated into train and test parts and we plan to use k-fold cross validation in order to come up with a good model for prediction.

References

[1] "Algorithm Competition." *Algorithm Competition*. N.p., n.d. Web. 28 Jan. 2017.