**Predicting Taxi Driver given Trajectory for the Day**

Yash Patel

Big Data Analytics, Worcester Polytechnic Institute, [yppatel@wpi.edu](mailto:yppatel@wpi.edu)

## 1 Introduction

With large data being collected, it is important to gain insight into the data to extract any features and make predictions with the data. We have a large dataset of paths that 5 taxi drivers used in one city for 6 months. The data included was the longitude and latitude of the driver at a certain time, the specific time, and whether the driver was currently driving someone around. Given this information, we would like to predict which driver (plate number) when given the trajectory of one taxi for the day.

## 2 Proposal

To achieve this prediction, I would like to train a neural network that when given a trajectory for the day would give the plate number of the driver. A trajectory is a list of data points. One data point contains the longitude, latitude, time, and status. The status is whether the driver was seeking for a passenger or currently serving a passenger. The goal is to extract overall performance for a trajectory and meaningful information from each data point.

## 3 Methodology

### 3.1 Data Processing

I created two zip folders for training data and validation data. The training data is a folder of all of the csvs. I would read each of the files and split the file into 5 trajectories, one for each driver. For each of the trajectories, I created a feature set. This would mean that the validation dataset would contain 25 results.

### 3.2 Feature Generation

Given a trajectory I split it into seeking, serving, and overall data. The reasoning for this is to keep information about the taxi driving throughout the day as well as obtain information about what the taxi did during the entire day. I also split the data into seeking and serving to see whether there were different patterns depending on whether the taxi drive was serving a passenger or not.

The seeking and serving data had four points: grid, hour, minute, second.

For each longitude and latitude, I calculated what grid they could be in. A grid cell would be a 1 km x 1km that would encapsulate all of the points that would fit in this. Each grid cell would need its own unique identifier. To calculate this I used the longitude and latitude to get a main point. Since longitude and latitude encapsulate a larger space i had to convert it into a 1km. This can be done by multiplying by 100 and flooring it. Then to make the number smaller so that the model can understand it better, we can subtract the smallest long and lat. To make it a unique identifier, we can combine the numbers with the longitude being the whole number while the latitude being the decimal.

Here is how I would do it:

114.088852 -> 1140.88852 -> 11408

15.7324312 -> 157.324312 -> 1573

Subtract the minimum:

11408 - 10000 = 1408

1573 - 1500 = 73

Then I would combine them:

1408.73

The hour, minute, and second can be extracted from the time column. This is simple since one time can look like this: 2016-07-01 00:00:03.

These 4 features would be one data point inside of the seek and serve features. The seek and serve features would be an array of these features to show time progression of how the taxi was moving.

Seek:

[[1408.753, 0.0, 10.0, 38.0], [1408.753, 0.0, …

Serve:

[[1411.753, 0.0, 0.0, 3.0], [1411.753, 0.0, 0....

The size of these features would be different because each taxi would be serving and seeking a different amount every day. If there was no serving data I make it so that it would contain only one data point of [0,0,0,0]

There are a total of 12 overall features that can be extracted from the entire day of one trajectory. These are :

[month, day, mostVisitedGrid, timesVisited, numGridsVisited, startHour, startMinute, startSecond, endHour, endMinute, endSecond, numTrips]

The month and day I got from the time column.

I found the most visited grid in a trajectory for both seeking and serving. I also included the count as the timesVisited

I also counted the number of unique grids driven to show how wide of a range a taxi driver would go.

The start time and end time were calculated by taking the time when the first time a driver started serving and the last time. The time is calculated by taking the number of minutes in the day.

The Number of Trips is how many trips a taxi went on. I thought of the trajectory as blocks of seeking and serving. So every time a taxi started serving after seeking that would be a new trip.

### 3.3 Network Structure

The model I came up with includes two lstm and a linear model. The two lstm are for the seeking and serving data. I then take the output from those two models and concatenate it with the overall data model output. It then goes through a few more linear layers to get to a result.

The LSTM model:

Linear (4 -> 10) (4 is how many features there are)

LSTM (10 -> 20)

Linear (20 -> 10)

Overall model

Linear (12 -> 30) (12 is the number of overall features)

Combined Model

Linear (10 + 10 + 30 -> 120)

Linear ( 120 -> 40)

Linear (40 -> 5)

Each of the linear layers is followed by a relu activation function because that was the most effective way to get better accuracy. The final layer is followed by a softmax function and the max of the softmax function is the resulting taxi.

I used Cross Entropy loss as well as the Adam optimizer.

### 3.4 Training and Validation Process

I split my data into 70% training and 30% testing. I also had a separate validation set that contained 5 unused days. I trained the dataset with a few different hyperparameters

Learn\_rate = [0.00001, 0.00002, 0.00005]

Epochs = [25, 100, 200]

I attempted using larger epochs but it made it so that my loss would never change.

## 4 Evaluation and Results

Accuracies:

| Learning Rate | Epoch | Accuracy |
| --- | --- | --- |
| 0.00001 | 25 | 0.3488 |
| 0.00002 | 25 | 0.2751937984496124 |
| 0.00005 | 25 | 0.43023255813953487 |
| 0.00001 | 100 | 0.3488372093023256 |
| 0.00002 | 100 | 0.2751937984496124 |
| 0.00005 | 100 | 0.43023255813953487 |
| 0.00001 | 200 | 0.32558139534883723 |
| 0.00002 | 200 | 0.4263565891472868 |
| 0.00005 | 200 | 0.18992248062015504 |

As you can, see my accuracy fluctuates heavily depending on the epochs and learning rate. My baseline was without the lstm structure and this is doing much better since with those models I was getting accuracies of 0.18. From further analysis I realized that the model was only outputting the same taxi. But with this model I was able to get more diverse set.

I’ve chosen the model that uses 0.0005 and 100 epochs since that gave the highest accuracy.

## 5 Conclusion

In conclusion, I was able to develop a model that has about a 0.4 accuracy on correctly guessing the taxi given its trajectory for the day. The preprocessing of data will extract the features of the trajectory and will separate them into seeking and serving. It can use both of these data points in the model to generate a prediction.