Forecasting San Francisco Parking Turnover Rate Using Meter Data

Maximillian Alfaro, Lance Fernando, Byron Han, Jacques Sham

1. Overview

Our project lives in the context of smart cities. A *smart city* is a city that utilizes data to manage its assets and resources more efficiently. Navigating such data management is an emerging research area. With the advancements in algorithms and the increasing availability of computing power, many companies and researchers have begun to employ machine learning techniques to solve smart city related problems.
A primary concern for smart city development is controlling traffic efficiently. A central subset within traffic management is parking management. In this paper, we will be focusing on a common metric used by cities to assess their parking efficiency: *parking turnover rate*. We will utilize cutting-edge distributed computing along with machine learning to forecast real-time *parking turnover* in hopes to lead cities towards an IoT infrastructure that best regulates parking management.

2. Background

What is parking turnover rate? "Parking turnover is an indicator of the rate of use of a parking space and the average number of vehicles using a given space or group of spaces during a specified time period. The turnover rate is determined by dividing the total number of vehicles parked in a given location by the capacity" [1].
Turnover rate is often used [2] as a metric for assessing the availability of parking spaces in a given area. Based on a variety of city parking analysis papers, we believe that turnover rate is positively correlated with the chance of finding parking; drivers have a greater chance of finding a vacant parking space in a given street block that has a high turnover rate and vice-versa.

3. Assumptions

The ability to analyze parking will be proportional to the quality of the data. Our data involves readings from parking meters. While this data is rich, we must acknowledge its limitations and in turn make some simplifying assumptions in order to proceed in our analysis.
In a perfect world, a driver would pay for the spot that they park in. In this same world, each driver would stay in their allocated parking spot for the duration of their paid period, either adding money or leaving once their payment period has ended. In this paper, we must make assumptions to simplify our analyses:

1. We assume that only one car parks in a spot for a single meter transaction
2. We assume high turnover rate for street parking increases the chance of a driver's ability to find a parking spot.

4. Data

1. SFMTA Parking Meter Detailed Revenue Transactions (Transactions) 7.9GB
   Contains all parking meters transactions within San Francisco
2. Parking Meter Information (Meters) 2.7MB
   Contains a description of every parking meter in San Francisco
3. Meter Rate Schedule (Schedules) 2.7MB
   Contains the pricing rate schedule of every parking meter in San Francisco

5. Tools

We first stored our data in AWS (Amazon Web Services) S3 (Simple Storage Service) bucket. We then set up a Mongo Database using multiple EC2 (Elastic Cloud Computing) instances, populating the Mongo Database with the data stored on S3. Finally, we launched AWS EMR (Elastic Map-Reduce) instances for machine learning with parallel computing by using tools Python, PySpark, Spark SQL, and Spark ML.

5a. MongoDB

Our MongoDB was set up using replication+sharding to distribute our data into collections. The configuration details can be found in the table below.

EMR configuration for Mongo

Server Type	**Instance Type**	**# of Replicas**	**CPUs**	**Memory**
Shard 1	t2.large	3	2	4
Shard 2	t2.large	3	2	4
Configuration Server	t2.large	3	2	4
Mongos Server	t2.large	1	2	4

5b. EMR

Elastic MapReduce (EMR) is simply a collection of EC2 instances managed by Hadoop that allows for easy manipulation of a large amount of data. We began with only three nodes – one master and two cores. Our data processing was performing more slowly than we would like, so we added an additional three task nodes.

EMR Specifications

Node Type	**Size**	**Instances**
Master	m4.xlarge	1
Core	m4.xlarge	2
Task	r5.xlarge	3

6. Methodology

We began by uploading our data to S3. We then built a distributed MongoDB over ten sharded EC2 instances. Tying this all together was an EMR cluster with six EC2 instances, from which we could access a Jupyter Notebook where we could perform analysis via Spark.

6a. Data Processing & Feature Engineering

We began by reading in our three datasets from MongoDB into three PySpark dataframes. We limited our scope to only focus on October 2017 - October 2018 which dropped our main dataset *Transactions* to

3.8GB. We are primarily interested in street blocks instead of specific meters so we aggregated up and grouped by street block in the *Transactions* dataset. For each street block, we calculated various summary statistics such as *NumTransactions, AvgDuration, etc.,* and most importantly *TurnoverRate* (response variable). This resulted in a dataframe where each row belongs to a street block, per day, per hour and their aggregated transaction info as well as meta info about the meters themselves on that particular block.

6b. Machine Learning Methodology

We began by clustering similar blocks based on their meta-data (i.e., number of meters, meter types). This allowed us to use the cluster labels as a feature instead of one-hot encoding instead of one-hot encoding all 1,300 *STREET_BLOCK*s. We then used SparkML to fit a decision tree with *maxDepth=5*.
wh

7. Analysis

Speed

Task	**Time**
Loading Data from Mongos	30 seconds
Feature Engineering	30.2 minutes
Training	5 minutes
Testing	1.3 minutes

Performance

Metric	**Value**
RMSE	9.005

8. Conclusion & Further Research

8a. Findings

Engineering the full data pipeline from S3 -> MongoDB -> EMR is a painstaking but highly beneficial process. Data processing was much more efficient using EMR as opposed to running locally.

With a low RMSE of 9.005, we are able to predict the *Turnover Rate* within 9 cars per hour for a particular block in San Francisco.

8b. Further Research

Although there are many possible roads leading out from this project, one interesting option would be visualizing the different clusters of blocks. Doing so could answer questions such as *What areas in San Francisco are similar in terms of finding parking?*
Other user-oriented use cases are abundant coming from this project. Perhaps an app that could be configured to give drivers real time parking information could be extended from this project. Maybe a

user-oriented data exploration interface in D3 would be interesting, giving the user the power to explore the San Francisco parking scene.
Although research about the legitimacy of turnover acting as an indicator of ease of parking is abundant, it seems not very analytical. An interesting prequel to this project would be an intensive analysis determining the ability of turnover-rate to inform the ease of parking for drivers.

9. References

[1] Downtown Parking Survey, City of Waupaca, Wisconsin
[2] Downtown Parking Management Study, City of Livermore, CA