ParkiTech: A Parking Lot Recommendation System

Yingfei Chen
yingfeichen@gatech.edu

Tianhao Wang
twang606@gatech.edu

Yuechen Wu
ywu788@gatech.edu

Winnie Zheng
yuewenz@gatech.edu

Jincheng Zhu
jzhu411@gatech.edu

1 INTRODUCTION

Nowadays, finding a parking spot has become a huge problem in daily life, especially for those who live in metropolis. Based on the research of Department of Parking and Traffic, for per square mile, San Francisco contains more cars than other cities in the United States [9], which makes parking situation even worse. Therefore, we find it important to build a user-friendly parking lot recommendation system to facilitate parking in urban areas. We believe that our design, which considers walking distance, parking cost and parking spot availability as a whole, has some unique edges over the currently available products.

2 PROBLEM DEFINITION

Our project aims to solve the parking spot finding problem in San Francisco. We want to design a user-friendly parking recommendation system to help drivers find the desirable parking spots near their destination.

3 SURVEY

The technical difficulties can be broken down into 3 main problems: time series analysis, geographic data aggregation, and risk-distance-cost trade-off problem.

3.1 Time series analysis

Consider the prediction of parking spot number as a time-based issue. Sun et al [10]. used a local linear regression model to predict short-term traffic based on the traffic speed data of Houston's freeway. Deshpande and Bajaj [3] discussed the implementation of the traffic flow prediction model using SVM based on the traffic data obtained near the Perungudi toll plaza in the corridor in Chennai, India. Chen et al [1]. and Hong et al [5]. both combined GA and support vector regression to predict the tourism flow. These papers provide us with a large number of machine learning models in traffic timing series prediction. At the same time, deep learning also proves helpful in time series prediction. Pflugler et al [7]. used a neural network to predict parking space availability in urban areas of Munich based on various factors. Chen [2] predicted parking occupancy in San Francisco using neural networks, ARIMA, linear regression, and support vector regression. It is found that the neural network provides the best prediction results among the models above. Haviluddin et al [4].,Purnawansyah et al [8]. and Wang [13] all used BPNN to forecast daily traffic and achieved excellent predicted results. However, Tavafoghi et al [11]. compared LSTM recursive neural network with the SARIMA model they proposed. They argued that the model-free model, like the LSTM, al- though results in a better prediction performance for short forecast horizons, may deteriorate much faster than that does the model-based method. Similarly, in Zhao et al.'s article [14], SVM outperformed the neu- ral networks in predicting the hourly occupancy of a single parking lot. Hence, the use of deep learning also requires caution and comprehensive comparison.

3.2 Geographic data aggregation

The functional division of cities is often regional. Parking demand in different urban functional areas has different patterns. Hence, it is possible to classify parking lots by area, to exploit geographical aggregation effects to obtain clearer data patterns and improve prediction accuracy. Chen [2] revealed the effect of aggregation on prediction for parking occupancy and used the k-means method to divide up the San Francisco city into seven regions. The clustering not only improved the accuracy of a single parking lot occupancy prediction but also had better interpretability. Similarly, Lin et al [12]. also segmented the on-street parking sensors data of Santander, Spain, into four different regions and developed neural network models for each region separately.

3.3 Risk-distance-cost trade-off

Several important factors should be taken into consideration when making a recommendation for parking. The trade-off between walk distance from the parking spot to the destination, cost of parking, and the risk of parking spots actually being fully occupied should be scientifically quantized. Landry et al [6]. used the expectation of searching time to measure the cost of finding an empty spot. Similarly, we could use the expectation of the time required to find a new spot over the risk of the spot being occupied to measure the risk. Walk distance and cost could also be measured in the same way and be assigned different weights.

4 PROPOSED METHOD

4.1 Intuition

4.2 Data Preprocessing

The original dataset (On-street parking data in San Francisco) contains the records of the measured parking availability in San Francisco, obtained from the public API of the SFpark project. In addition, by combining the SFpark data with taxi trajectories from the Cabspotting project, we simulated parking crowd-sensing with taxis as probe vehicles. Each line of the dataset corresponds to the parking situation of a parking segment in the SFpark pilot area at a specific time stamp. The columns correspond to the following content:

- **timestamp**: Timestamp of the API record rounded to the closest minute.
- **segmentid**: ID of the parking segment (correspond street name and coordinates also available in another data file).
- **capacity**: Current total number of parking spaces in the parking segment.
- **occupied**: Current number of occupied parking spaces in the parking segment.
- **observation***K*: The number of occupied parking spaces in the parking segment observed by probe vehicle K. In each line of the dataset, 10 observation values are included.

In total there are $5,080,320$ lines of data in the original dataset. These include the observation values of 420 parking segments (abbreviated as parking lot) in June and July 2013. According to our observation (shown in Figure 1), the parking lots are located in roughly 9 areas in San Francisco (marked with different colors). As suggested by X. Chen, parking lots in different regions subject to different change patterns, we first used K-means clustering to split the original dataset to 9 files (each for one cluster). This also allows us to extract more features from the split dataset.

In order to make recommendations of parking lots, we

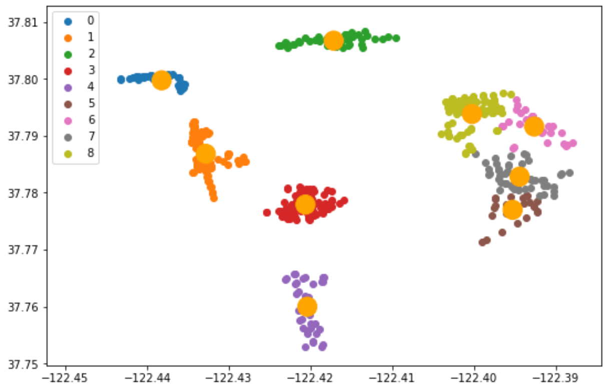


Figure 1: Clustering of parking spots

reconstructed 3 datasets, including *segment data*, *segment info* and *center info*. *segment data* dataset includes the occupancy of parking lots and time, indexed by segment ID. segment ID is also used to find the information of the parking lot in *segment info*, including the cluster it belongs to, coordinates and capacity. *center info* stores the coordinates of the centers of the 9 clusters. When using these 3 datasets for training and testing, we combined them together and select 7 features for parking spot number prediction, including

- **cluster**: The number of the cluster a given parking spot belongs to. The value is an integer between 0 and 8 (inclusive).
- **x**: The x coordinate of a given parking spot. e.g. -122.4164831.
- **y**: The y coordinate of a given parking spot. e.g. 37.80722541.
- **time**: The time when the data is recorded, rounded to hour. For example, if a line of data is recorded at 13:31 PM, the time will be set as 13. The value is an integer between 0 and 23 (inclusive).
- **weekday**: The time when the data is recorded. For example, if a line of data is recorded on Monday, the weekday will be set as 1. The value is an integer between 0 and 6 (inclusive, Sunday as 0).
- **occupied**: The number of occupied parking spot. This value has to be smaller than capacity. Since the number is observed by multiple probe vehicles and we took the mean of their observations, this value could be decimal.
- **capacity**: The number of all parking spots in a given parking lot.

When we plot the ratio of **occupied** parking spots with respect to total **capacity** over the map (shown in Figure 2), we found that the situation in different clusters varies greatly, which also justifies our previous application of clustering method. Even in the same cluster, the distance to the center could also influence the ratio. Hence we further computed a higher-level feature **distance** based on the coordinates **x**, **y** of parking lots and the center of **cluster**.

Also we plotted the ratio of **occupancy** with respect to **capacity** change within 24 hour based on data of some parking lots (shown in Figure 3). We can almost be sure that there exists a pattern in the change of ratio. Also, presumably the parking situation could vary on different weekdays. For example, there could be more cars parking in front of a grocery during weekends than on Monday. Therefore, we also took **time** and **weekday** into our consideration.

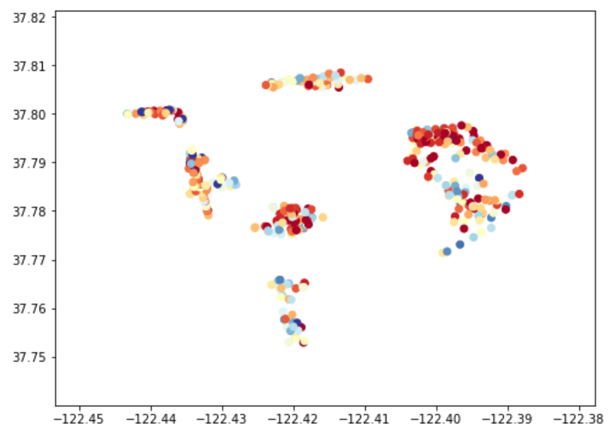


Figure 2: Heatmap of average occupancy ratio of parking spots at 12PM everyday

Last but not least, the predicted **occupied** parking spot number should be no larger than **capacity**. Thus, we limited the predicted **occupied** parking spot number with **capacity**. Through data preprocessing, we managed to compress the original dataset to new datasets with 423, 360 lines of data belonging to 420 parking lots.

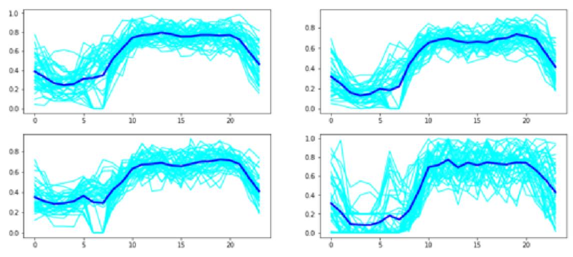


Figure 3: Ratio of *occupancy* with respect to *capacity* change within 24 hours of 4 randomly selected parking lots

4.3 Prediction

We used linear regression model to predict the number of **occupied** parking spots. Consider that the availability of parking spots over the past 24 hours could be a good reference while predicting the current number, we included these 24 values into the features. Also, as mentioned before, **time** (hour), **weekday** and **distance** were also taken as features. Note that we consider the situation in different clusters separately, we did not include the cluster number into the features. Hence, we performed linear regression over the 27 features listed below.

- **occupied**H: The number of occupied parking spots during the Hth hour among the past 24 hours.
- **distance**: The distance from a given parking lot to the cluster center.
- **time**: The hour timestamp of a given data plus the estimated travel time (rounded to hour).
- **weekday**: The weekday timestamp of a given data.

4.4 Recommendation

The goal here is to provide the recommendation for parking lot choice based on the estimated time of finding an available parking lot.

Firstly, lets define two functions. For two locations g_1 and g_2

$$w(g_1, g_2) = \frac{\text{Distance between } g_1 \text{ and } g_2}{\text{Walking speed}} \tag{1}$$

$$d(g_1, g_2) = \frac{\text{Distance between } g_1 \text{ and } g_2}{\text{Driving speed}} \tag{2}$$

Then We use a value iteration method to estimate the expected total time for each parking lot.The total time consists of two parts, driving to a parking lot to find an empty place and then walk to the destination. The other part is driving to a parking lot which is full then rediscover and head to the next most suitable parking lot. As long as we get the parking time cost estimation (PTCE). We recommend the parking lots base on minimum PTCE (Min_PTEC) rule.

4.5 User Interface Design

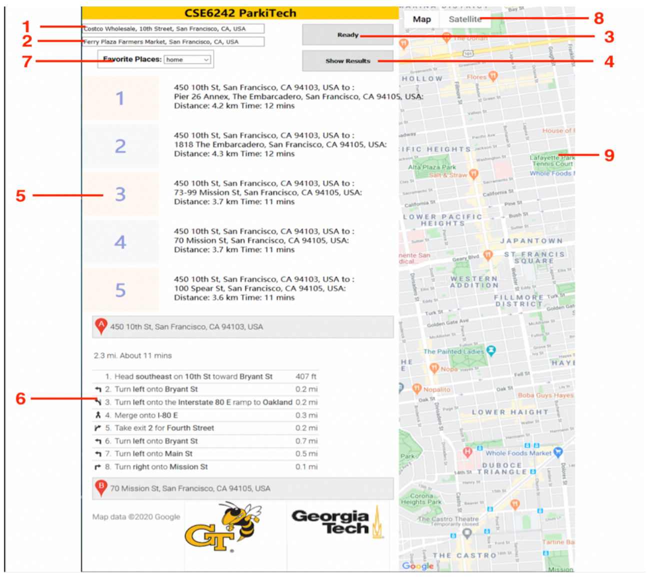
The principle of our design is to achieve functionality and user-friendliness at the same time. We hope

Algorithm 1: Parking Time Cost Estimation (PTCE)

```
Input:
  Current location: $g_x$ ,
  Destination: $g_y$,
  Parking lot locations (in certain range):
  $g_n, n = 1, 2 \dots N$,
  Probability of each parking lot being vacant:
  $p_n, n = 1, 2 \dots N$ ,
  Iteration times M,
  Penalty for not found: $\alpha$,
  Penalty for moving to another parking lot: $\beta$
Output: The cost for each candidate parking lot
        after M iterations: $c_M(n), n = 1, 2 \dots N$
1 $c_n = p_n * w(g_n, g_y) + (1 - p_n) * \alpha$   // Initialize the cost
2 for $i$ = 1,2...$M$ do
      /* Update $c_i(n)$ based on $c_{i-1}(n)$ */
3     for $n$ = 1,2...$N$ do
          /* Update the cost considering the next best choice when nth parking lot is full */
4         $c_i(n) = p_n w(g_n, g_y) + (1 - p_n) \min_{m \neq n}(d(g_m, g_n) + c_{i-1}(m) + \beta)$
5 $c_M(n) = c_M(n) + d(g_x, g_n)$   // Add the time driving from current location
```

users can easily start even at the first glance and enjoy the Google Maps-style user experience. Therefore we carefully arranged the layout of panels and buttons, considered the logistic relations between each element, and employed the most stable and fastest method to accomplish server requests. Based on such considerations, we came up with the simple yet powerful system design shown in Figure 4. Detailed deception of the panels and buttons and their functions can be found in section 5.3.

Beyond front-end layout design, we also found a practical solution to address the connection problem between front-end and back-end. When users input their address from the front-end, the request will be transmitted to the python scripts of back-end. Executing Python scripts will generate a CSV file containing the detailed information of recommendations, including geographic coordinates, predicted time cost and parking space availability. Then, the front-end uses D3.js to load data and complete the following series of operations. In order to run the scripts when clicking a button



Costco Wholesale, 10th Street, San Francisco, CA, USA
Ferry Plaza Farmers Market, San Francisco, CA, USA
Favorite Places: home

1 450 10th St, San Francisco, CA 94103, USA to :
Pier 26 Annex, The Embarcadero, San Francisco, CA 94105, USA:
Distance: 4.2 km Time: 12 mins

2 450 10th St, San Francisco, CA 94103, USA to :
1818 The Embarcadero, San Francisco, CA 94105, USA:
Distance: 4.3 km Time: 12 mins

3 450 10th St, San Francisco, CA 94103, USA to :
73-99 Mission St, San Francisco, CA 94105, USA:
Distance: 3.7 km Time: 11 mins

4 450 10th St, San Francisco, CA 94103, USA to :
70 Mission St, San Francisco, CA 94105, USA:
Distance: 3.7 km Time: 11 mins

5 450 10th St, San Francisco, CA 94103, USA to :
100 Spear St, San Francisco, CA 94105, USA:
Distance: 3.6 km Time: 11 mins

A 450 10th St, San Francisco, CA 94103, USA

2.3 mi. About 11 mins

1. Head southeast on 10th St toward Bryant St	407 ft
2. Turn left onto Bryant St	0.2 mi
3. Turn left onto the Interstate 80 E ramp to Oakland	0.2 mi
4. Merge onto I-80 E	0.3 mi
5. Take exit 2 for Fourth Street	0.2 mi
6. Turn left onto Bryant St	0.7 mi
7. Turn left onto Main St	0.5 mi
8. Turn right onto Mission St	0.1 mi

B 70 Mission St, San Francisco, CA 94105, USA

Map data ©2020 Google

1. Enter Origin: Input the start location
2. Enter Destination: Input the final location where the user wants to find parking lot around
3. Ready: Send query to the backend to handle the user's request
4. Show Results: Display the suggestion list
5. Recommendation List: Five top parking spot near the destination
6. The Navigation: the detailed navigation from the start location to destination
7. Favorite Place: The favorite original location for user including gym, corporation, grocery store, and home
8. Satellite: Show the street view around the parking lot
9. Google Maps

Figure 4: The design of user interface of this system

on the webpage, out of consideration of safety, we can use Tornado to construct a local web server. Otherwise, users can also create a handler to listen to the request sent to a specific port like what we did, which is also a wise approach. This step is where the difficulty lies if the user would like to run the system demo personally on their PC. Figure 5 demonstrates how the connection between front-end and back-end works.

5 EXPERIMENTS

Our experiments covered all the key points mentioned above, which can be summarized as follow:

- Whether our prediction for the parking spot number is accurate.
- Whether the recommendation system could functionally provide users with a list of reasonable parking spots.
- Whether the interface is friendly and easy to access for users.

In order to answer these questions, we conduct the following experiments.

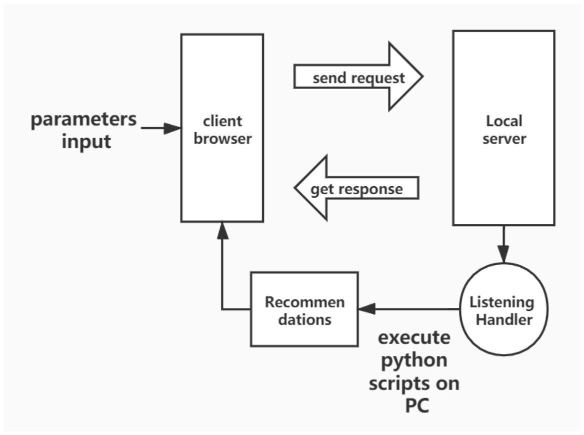


Figure 5: Connection between front-end and back-end

5.1 Prediction

As mentioned in section 4, we used linear regression to predict the number of occupied parking spots. We split the preprocessed dataset in to training and testing data, with the ratio of 19 : 1. Among the training data, we used $\frac{1}{19}$ for validation. When training the linear regression model, we used default settings of Scikit-learn. Figure 6 shows error rate of occupied parking spot number prediction, i.e. the difference between the predicted and true ratio of occupied parking spot number with respect to total capacity. The average error rate is only 0.0898 and the variance is 0.0125, which means that in most cases, the prediction error of occupied parking spot number is less than 1. Hence, our prediction is proved accurate and reliable.

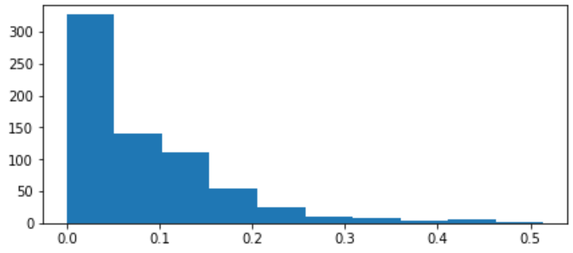


Figure 6: Histogram of error rate of occupied parking spot number prediction

5.2 Recommendation

First, let's start with an example to qualitatively and intuitively look at the internal logic of our algorithm. In the Figure 7, each dot in the figure represents a parking lot. The darker the color of the dot, the greater the number of predicted vacancies. The larger the dot, the higher the recommendation level. During initialization, considering each parking lot separately, the recommended level for the parking spaces with more vacancies is higher. 142 and 121 are significantly more recommended than 203. After iterating through our algorithm, considering the aggregation of the parking spaces in the upper right corner, parking space 203 has received higher recommendation.

This is reasonable. If you go to 143 without finding parking space, (although the probability is low) the surrounding 142, 126, and 127 have very few predicted vacancies (the pale ones can hardly be recognized in the picture), you will most likely need to run a lot of different places or even go quite far to find an empty parking space. But if you go to 203 and do not find one, you can go to the surrounding 131, 141, 121, 122 to continue searching, they are all parking lots with more vacancies.

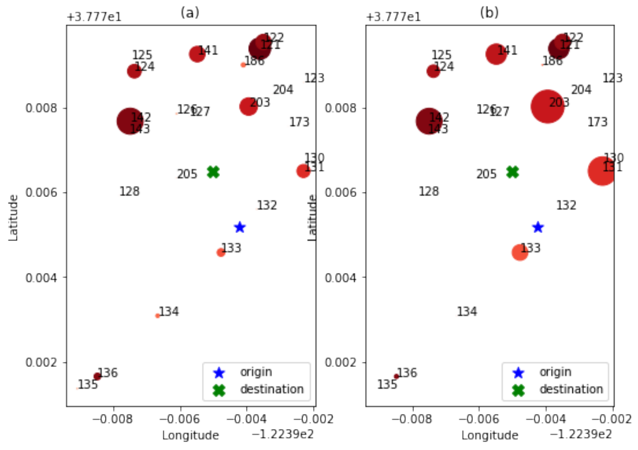


Figure 7: (a) Recommendation level initialization (b) Recommendation level after iteration

We test our decision rule Min_PTCE with common decision rule Min_Distance (always looking for the nearest parking lot). We compute the average time driving to find a parking space and walking to destination. The average time here takes into account the situation of going to a parking lot without finding empty spaces. Decision-making process to find the next parking space will not stop, until the cumulative probability of finding a parking space is expected to reach 99%. Each group of test has 50 uniformly samples with random origin, destination locations and advance time. (Location range: latitude 37.74 to 37.82 & longitude -122.46 to -122.38; Time range: 0 to 24h)

We can see that our decision-making method can steadily and significantly reduce the estimated time spent on parking compared with Min_Distance rule.

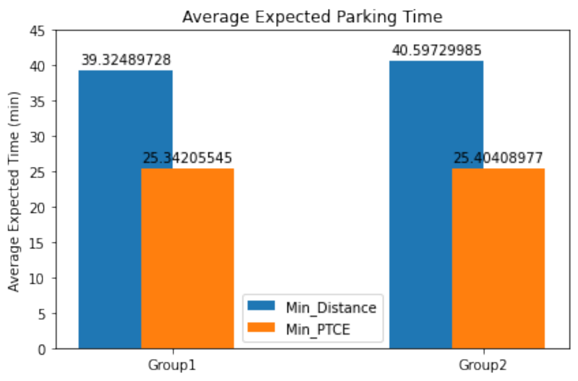


Figure 8: Average expected parking time

5.3 User-friendliness

We tested all the panels and buttons on the interface, including

- **Autocomplete input**: Thanks to the built-in association function inherited from Google Maps API, users don't need to enter absolutely accurate origin and destination address to the interface.
- **Simple yet functional interface design**: After entering origin and destination, users just need to click the button on the right and recommendation results will be displayed on the left panel, which includes the top five recommended parking locations. Specifically, detailed address, distance and estimated time will be shown. Users can choose their desired parking lot by clicking on the corresponding button then view detailed navigation information.
- **Favorite addresses**: Users can select the common origins from the drop-down box as origin. For example, if "grocery" is selected, our system will automatically generate its recommendation for the trip from the current location to the destination.
- **Satellite and street View**: To help knowing about street environment around destination in advance, we accessed our users to Google Maps satellite map and street view. Almost all Google Maps functions are inherited, which allows new users to start easily.

We entered the origin "pier 39" and destination "Oracle Park". It took around 2 seconds to show the results. We randomly picked one parking spot recommended, the route is displayed on the map once we click on the corresponding number button. Then we selected the saved location "gym". The system took it as the destination and take the (pre-set) current location as origin, and made an recommendation immediately. The entire operation is easy and smooth.

6 CONCLUSION

Our design - ParkiTech, which incorporates parking spot number prediction and the novel recommendation mechanism, contributes to solving the parking problem in San Francisco. Well tested by our comprehensive experiments, its functions can provide multiple recommendations for users to choose from, even before they arrive. Users can benefit to a great extent from the accurate predictions, smart recommendations and professional navigation services.

However, there is still room for improvements. For now our system is only applicable to the San Francisco area. In order to increase the flexibility, more detailed data of other regions is required to help us eliminate the regional restriction and benefit more people. Also, limited by the available dataset, real-time update of the data is another problem. Hence, we hope to introduce a "spot and report" function in the future, which allows users to upload the latest number of parking spots from the interface such that more accurate prediction models can be constructed. Last but not least, due to the limited computing capacity of our laptops, we simplified the algorithms and frameworks of our system. For example, the walking distance is computed using ℓ_2 distance instead of Manhattan distance. Existing parking limitation and free parking hours are neglected. Users have to wait a couple of seconds before the results can be shown. More enhancement can be made once we switch to server platforms.

For further development, our team will focus on redesigning the system for mobile devices. Since the computing capacity of smart phones is usually limited, optimization of our code and incorporation of cloud computation are required in order to provide better user experience. Promoting our system is also necessary such that more users and engineers can contribute to the long-term development.

APPENDIX I: DISTRIBUTION OF WORK

Task	Week	People
Data preprocessing	1,2	JZ
Construction of front-end	1,2,3	TW, YC
Prediction model design	5	JZ, YW
Trade-off quantization	5	YW, WZ
Assembly of front/back-end	6	TW, YC, YW, JZ
Overall test	6,7	All
Final report writing	7	All

Here we use abbreviation of our first name.

- YC = Yingfei Chen
- TW = Tianhao Wang
- YW = Yuechen Wu
- WZ = Winnie Zheng
- JZ = Jincheng Zhu

REFERENCES

[1] Kuan-Yu Chen and Cheng-Hua Wang. Support vector regression with genetic algorithms in forecasting tourism demand. *Tourism Management*, 28(1):215–226, 2007.

[2] Xiao Chen. Parking occupancy prediction and pattern analysis. *Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep. CS229-2014*, 2014.

[3] Minal Deshpande and Preeti Bajaj. Performance improvement of traffic flow prediction model using combination of support vector machine and rough set. *International Journal of Computer Applications*, 163(2):31–35, 2017.

[4] Haviluddin Haviluddin and Rayner Alfred. Daily network traffic prediction based on backpropagation neural network. 2014.

[5] Wei-Chiang Hong, Yucheng Dong, Li-Yueh Chen, and Shih-Yung Wei. Svr with hybrid chaotic genetic algorithms for tourism demand forecasting. *Applied Soft Computing*, 11(2):1881–1890, 2011.

[6] David MW Landry and Matthew R Morin. Estimating parking spot occupancy. 2013.

[7] Christoph Pflügler, Thomas Köhn, Maximilian Schreieck, Manuel Wiesche, and Helmut Krcmar. Predicting the availability of parking spaces with publicly available data. *Informatik 2016*, 2016.

[8] Purnawansyah Purnawansyah and Haviluddin Haviluddin. Comparing performance of backpropagation and rbf neural network models for predicting daily network traffic. 2014 Makassar International Conference on Electrical Engineering and ..., 2015.

[9] SFGate Article. *The high cost of free parking*, June 2005.

[10] Hongyu Sun, Henry X Liu, Heng Xiao, Rachel R He, and Bin Ran. Use of local linear regression model for short-term traffic forecasting. *Transportation Research Record*, 1836(1):143–150, 2003.

[11] Hamidreza Tavafoghi, Kameshwar Poolla, and Pravin Varaiya. A queuing approach to parking: Modeling, verification, and prediction. *arXiv preprint arXiv:1908.11479*, 2019.

[12] Eleni I Vlahogianni, Konstantinos Kepaptsoglou, Vassileios Tsetsos, and Matthew G Karlaftis. A real-time parking prediction system for smart cities. *Journal of Intelligent Transportation Systems*, 20(2):192–204, 2016.

[13] Peng Wang, Gang Zhao, and Xingren Yao. Applying back-propagation neural network to predict bus traffic. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 752–756. IEEE, 2016.

[14] Ziyao Zhao and Yi Zhang. A comparative study of parking occupancy prediction methods considering parking type and parking scale. *Journal of Advanced Transportation*, 2020, 2020.