Traffic Predicion With Advanced Graph Neural Networks

End-term Report

11812804 董　正
11813225 王宇辰
11811305 崔俞崧

Supervisior: 宋轩

Department of Computer Science and Engineering

Jan. 2021

Content

1 Preliminaries

1.1 Review

1.1.1 TTE

Travel Time Estimation (TTE) is one of the most important researching topic in the traffic forecasting field. Estimating the travel time of any path in a city is of great importance to traffic monitoring, route planning, ridesharing, taxi dispatching, etc. On Sep. 2020, DeepMind published a blog named *Traffic prediction with advanced Graph Neural Networks*. This blog briefly described the whole industrial structure of estimated times of arrival (ETAs) techniques applied in Google Map but did not given any detailed implementation or any code. Our work is based on the model structure of TTE proposed in the blog.

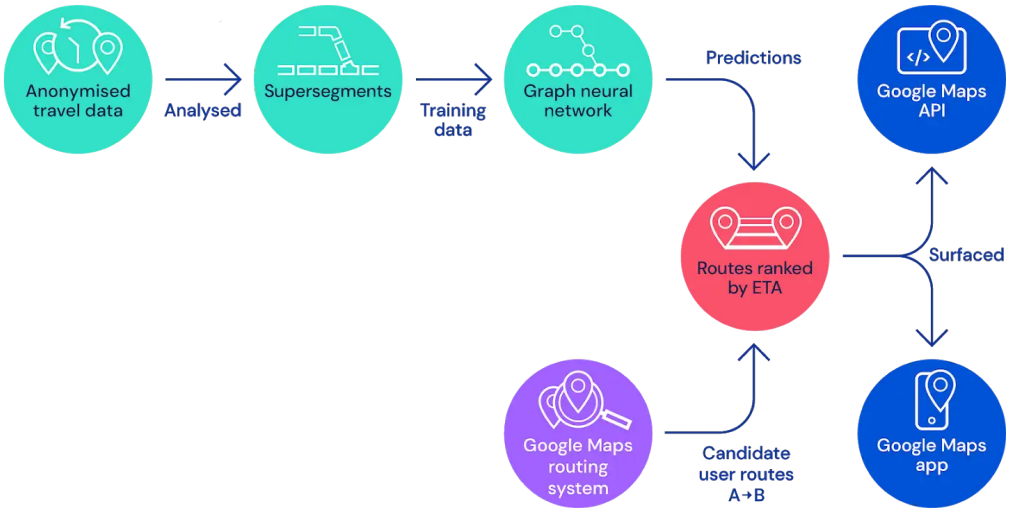


Figure 1: Architecture

1.1.2 Goal

Our ultimate goal (tentative) is to implement the industrial structure and apply it to the open source databases in China, then compare the performance with the state-of-the-art structures and find its application value. This semester, we will first learn about the relative algorithms. More specifically, the two main algorithm we will use: TTE algorithm and Graph Neural Network (GNN). After that, we will learn the methods of processing traffic data. Finally, we combine the algorithms and apply them on some open source database.

1.2 Introduction

In the last stage, we have got some open source data and learned how to process the data. And we got deep in the code of *DeepTTE* and researched on a new concept called **Travel Time Index (TTI)**. In addition, we proposed a new computing process of *Supersegment*.

Breifly, we will state our work in this report as:

- DeepTTE Result Analysis by 王宇辰
- Calculation of TTI by 崔俞崧
- An implementation of Supersegment by 董正

2 DeepTTE Result Analysis

Due to previous result on training loss, although training loss is level off to 0, test loss remians about 0.2, as shown in figure 2. I tried to change the hyperparameters, but cannot get an explicit better result. Then, a deep research on the result is applied.

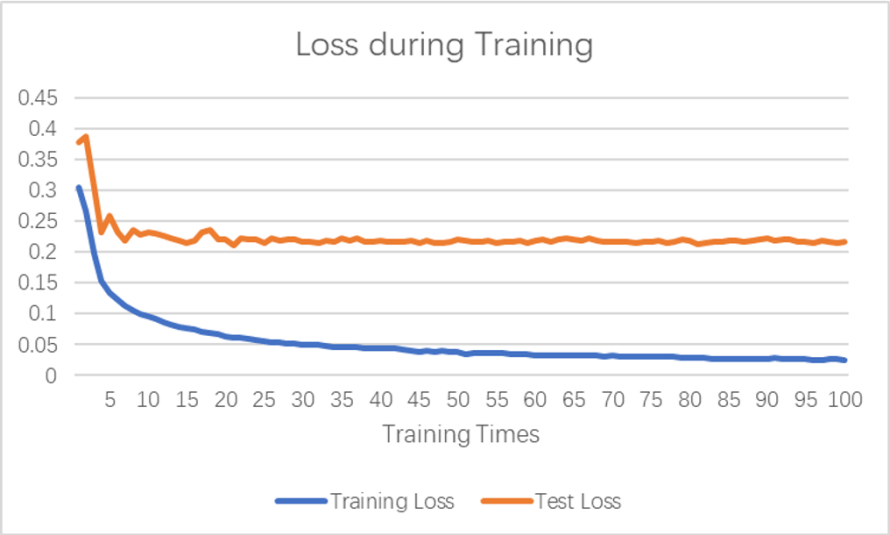


Figure 2: Total Loss during Training

2.1 The training process

As shown in figure 3, the training result is shown every 20 epoches. We can simply analyze by using $\frac{Predict\ Time}{Exact\ Time}$ to find the trend during training.

In the first epoch, shown in figure 3(a), the average ratio is 0.5, with great difference from 1.0.

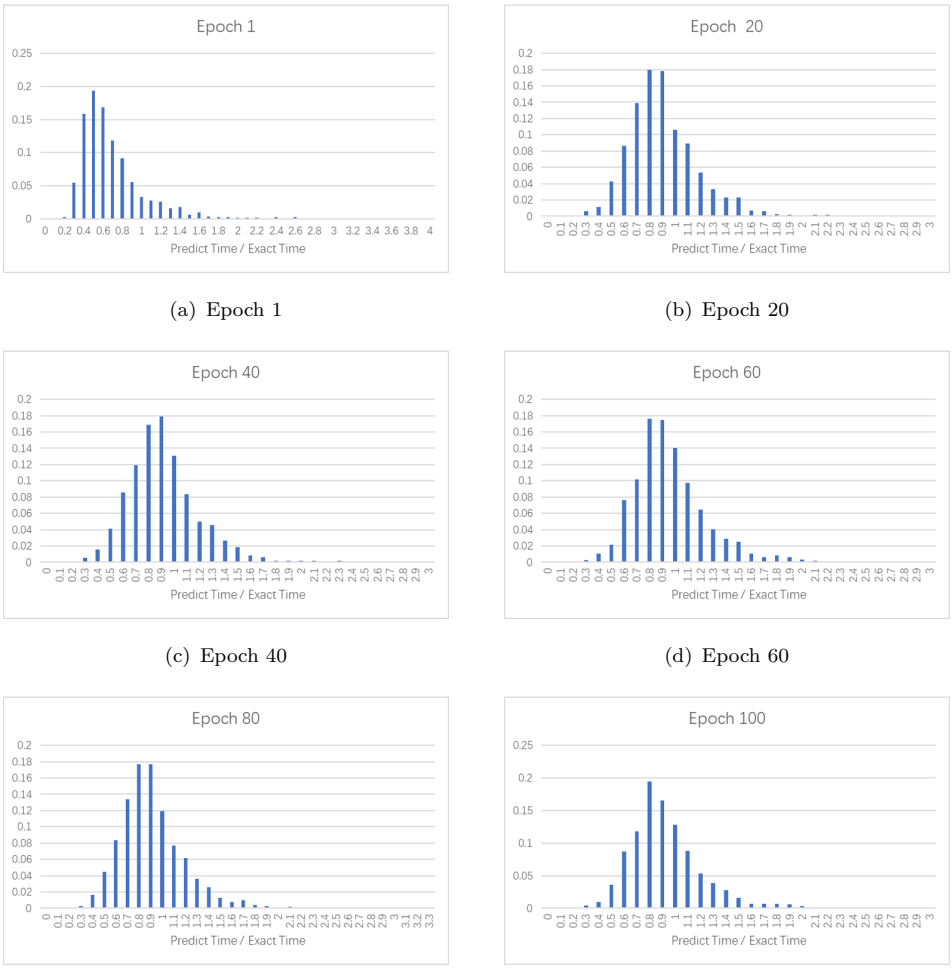
In the 20th epoch, shown in figure 3(b), the average ratio is 0.8, with distributed ratio.

In the 40th epoch, shown in figure 3(c), the average ratio is around 0.8 - 0.9, with ratio < 0.8 reduced and retio > 0.8 increased.

In the 60th epoch, shown in figure 3(d), the average ratio is 0.8, with ratio < 0.8 less than 0.2.

In the 80th epoch, shown in figure 3(e), the average ratio is around 0.8 - 0.9, with ratio > 0.9 reduced.

In the 100th epoch, shown in figure 3(f), the average ratio is 0.8, with gradually form an exponential function distribution.



(a) Epoch 1

(b) Epoch 20

(c) Epoch 40

(d) Epoch 60

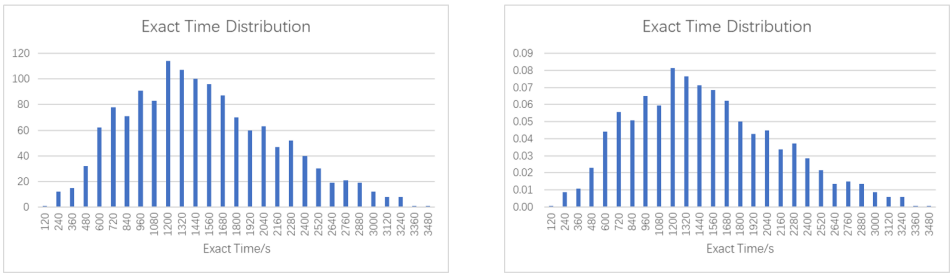
(e) Epoch 80

(f) Epoch 100

Figure 3: Total Loss during Training

2.2 The Exact Time

The exact time distribution is shown in figure 4, mainly within 480s - 2400s, about 8min - 40min. Thus, we choose to analyze every 10min, equal to 600s.



(a) In number

(b) In probability

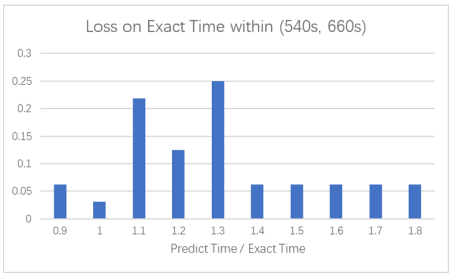
Figure 4: Exact Time Distribution

Around 600s, as shown in figure 5(a), Ratio is higher than 1.0, prediction is about 30% over exact time.

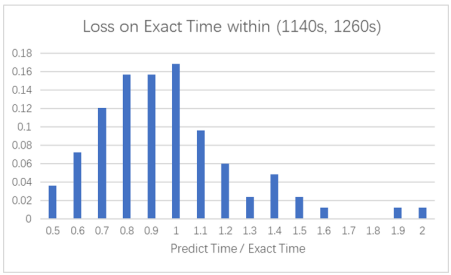
Around 1200s, as shown in figure 5(b), Ratio is about 0.9 - 1.0, prediction is most accurate around 1200s.

Around 1800s, as shown in figure 5(c), Ratio is around 0.9, prediction is about 10% below exact time, can be applied at all.

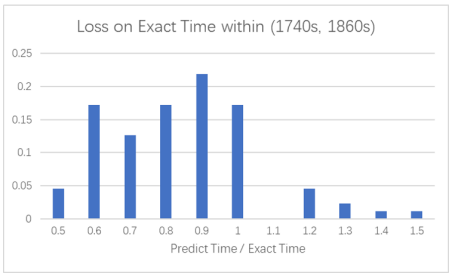
Around 2400s, as shown in figure 5(d), Ratio is about 0.6 - 0.9, prediction is inaccurate as it is about 30% difference in average.



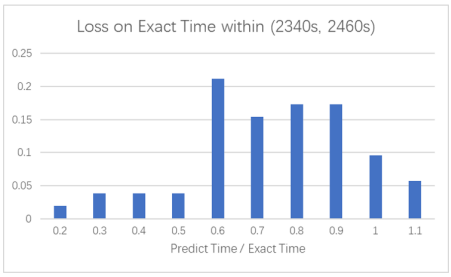
(a) Around 600s



(b) Around 1200s



(c) Around 1800s



(d) Around 2400s

Figure 5: Sample Distribution

2.3 Conclusion

Return to the question from figure 2, we can find out something.

- In the training process, the ratio is gradually centerlized to 0.8 - 1.0.
- Predict time around 1200s –1800s is accurate, about 10% error.

3 Travel Time Index

The TTI (Travel Time Index) industry's most used evaluation index of urban congestion degree is the ratio of the actual travel time and the free flow time.

$$TTI = \frac{\sum_{i=1}^{N} \frac{L_i}{V_i} \cdot W_i}{\sum_{i=1}^{N} \frac{L_i}{V_{free\_i}} \cdot W_i}$$

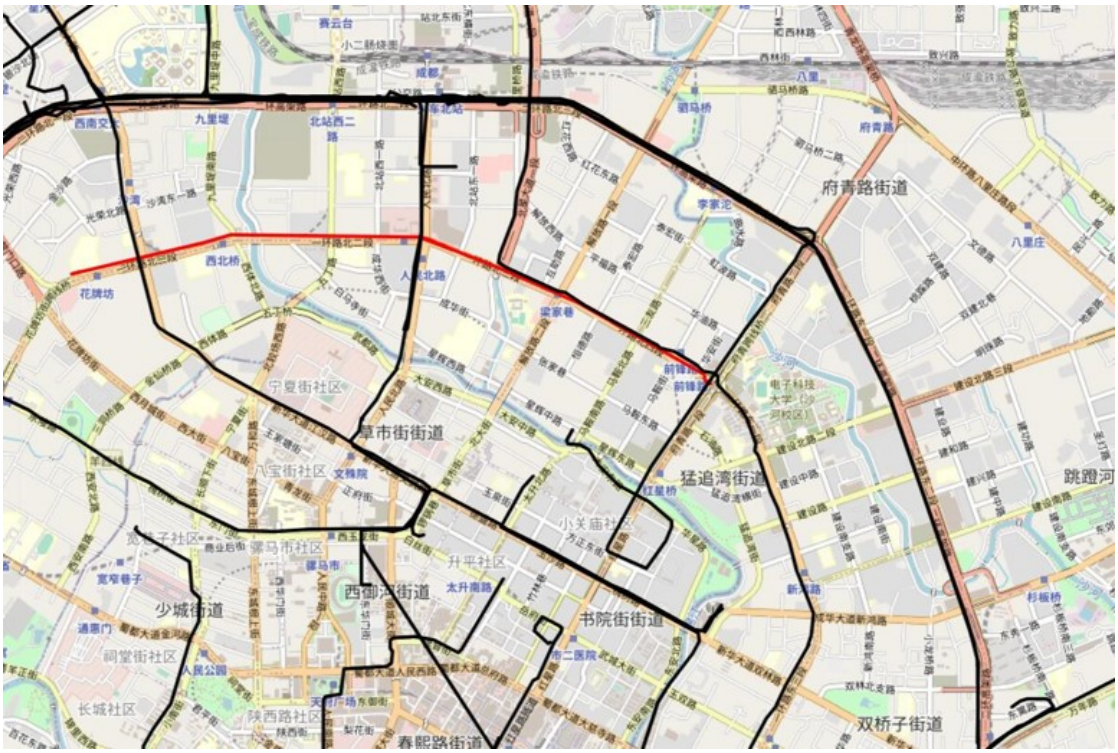
- L_i: length of the road
- W_i: weight of road
- V_i: real time traffic information
- V_{free_i}: free flow velocity

3.1 Dataset

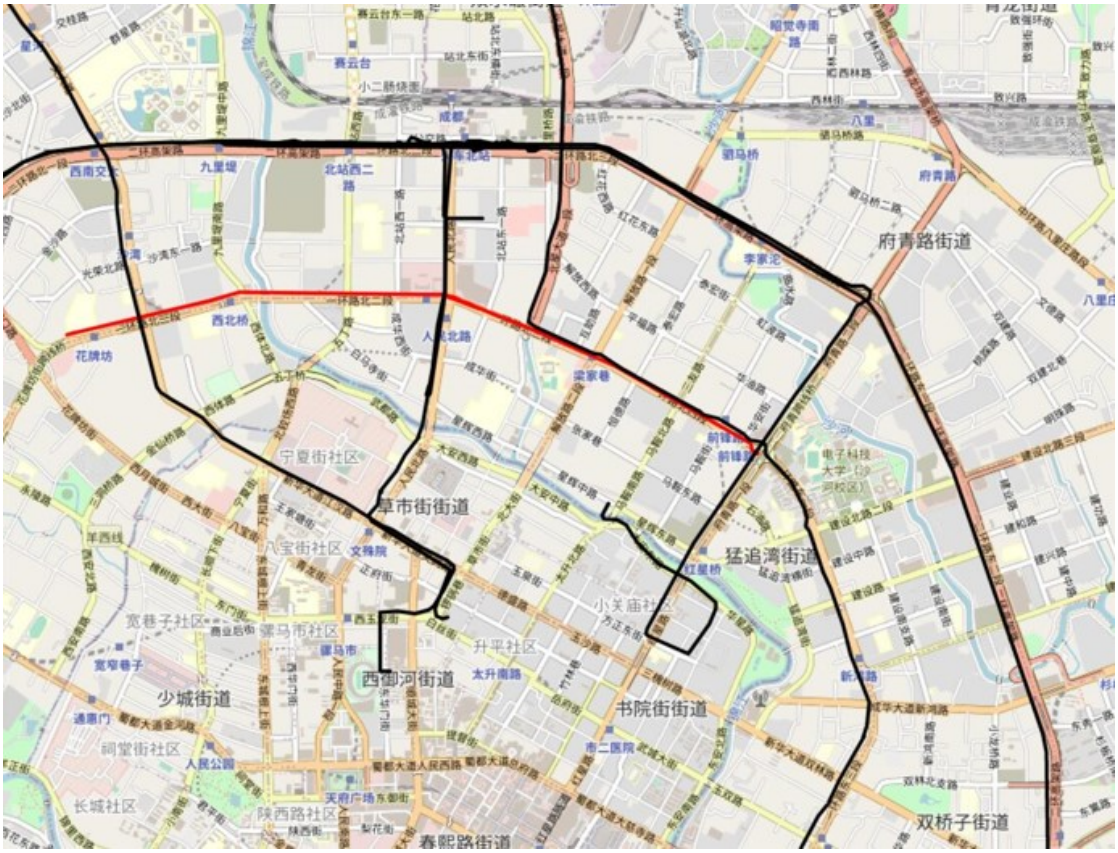
- Data source: Didi Chuxing GAIA Intiative
- Region: Chengdu
- Time: 2018-10-01 to 2018-12-01
- Content:
  - GPS track of taxis with timestamps
  - Coordinate of road and district boundaries
  - TTI and average speed of districts

3.2 Computing TTI

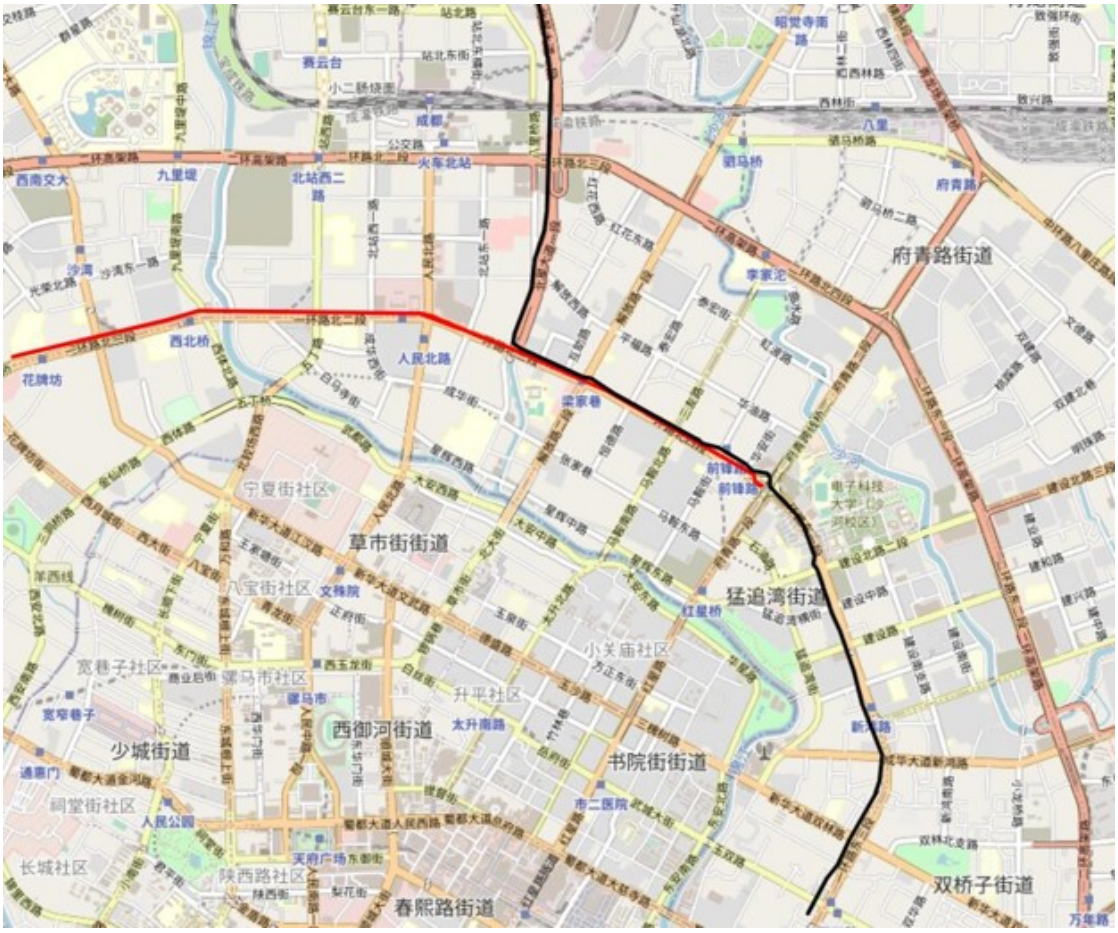
To calculate the TTI, the most important thing is to calculate the average speed of the section and the free flow speed. In order to get the average speed of the road section, it is necessary to calculate through a large number of track data. First, select the section of road to be calculated and the time (here we take the north of the first ring road.;10:00 a.m.-10:10 a.m. as an example), and draw all the tracks and sections as shown in the figure below. Note that in order to make the trajectories clearly visible, 20 trajectories are selected here for drawing demonstration. The number of trajectories in the actual calculation process is far greater than that shown in the figure. (The black line segment represents the vehicle track and the red line segment represents the road)



Then, the track is matched with the section. The unmatched track is removed, and only the successfully matched track is retained. The matching method is mainly carried out by judging the distance between the track point and the road section. Therefore, even if the track only intersects with the road section, it will be successfully matched here.



However, a track that intersects rather than overlaps with a section of road is considered a match, but this type of track does not reflect the traffic situation of the section at all. Therefore, the overlap between the successfully matched track and the road section is then calculated. If the overlap is too small, the track will be deleted to ensure that the successfully matched track is all the track that has been traveled in the road section.

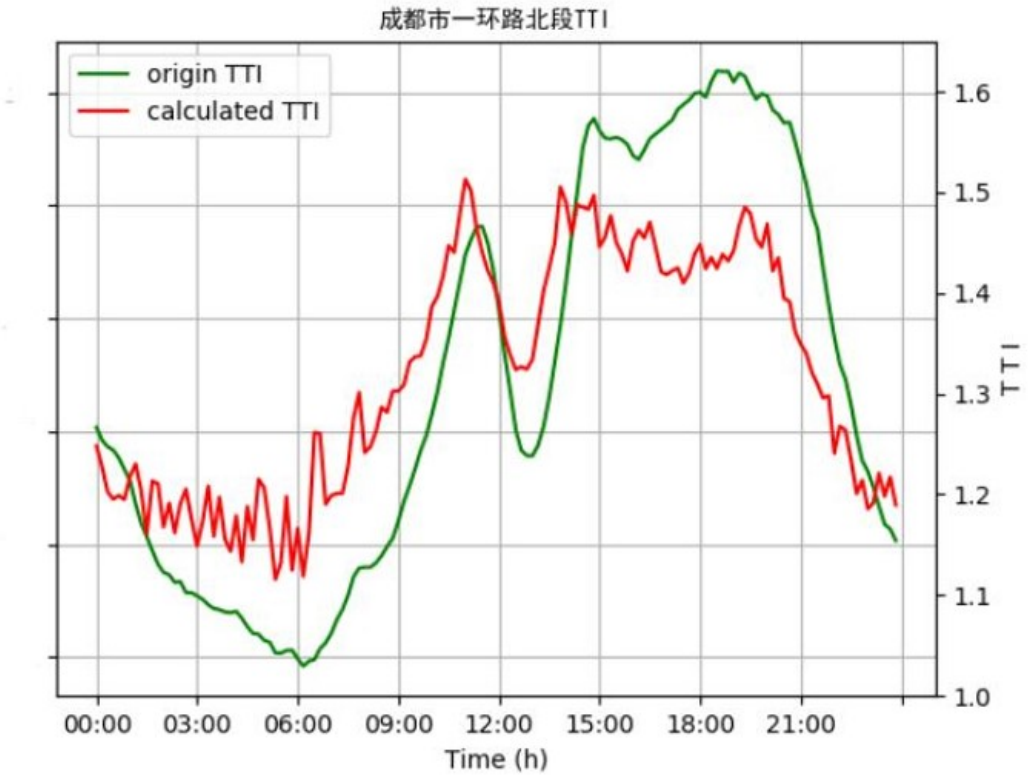


After data processing, the average speed of the section is calculated. Firstly, the processed track is used to calculate the average velocity of the track, which is set as V_t (t represents the number of tracks). After the average velocity of each track is obtained, the average velocity V_{avg} of this section is calculated as followed: (Here n represents the total number of matched sections)

$$v_{avg} = \frac{1}{n}\sum_{t=1}^{n} v_t$$

Secondly, the free flow velocity of the section is calculated. The speed between 4:00 and 4:30 is selected here as the free flow speed.

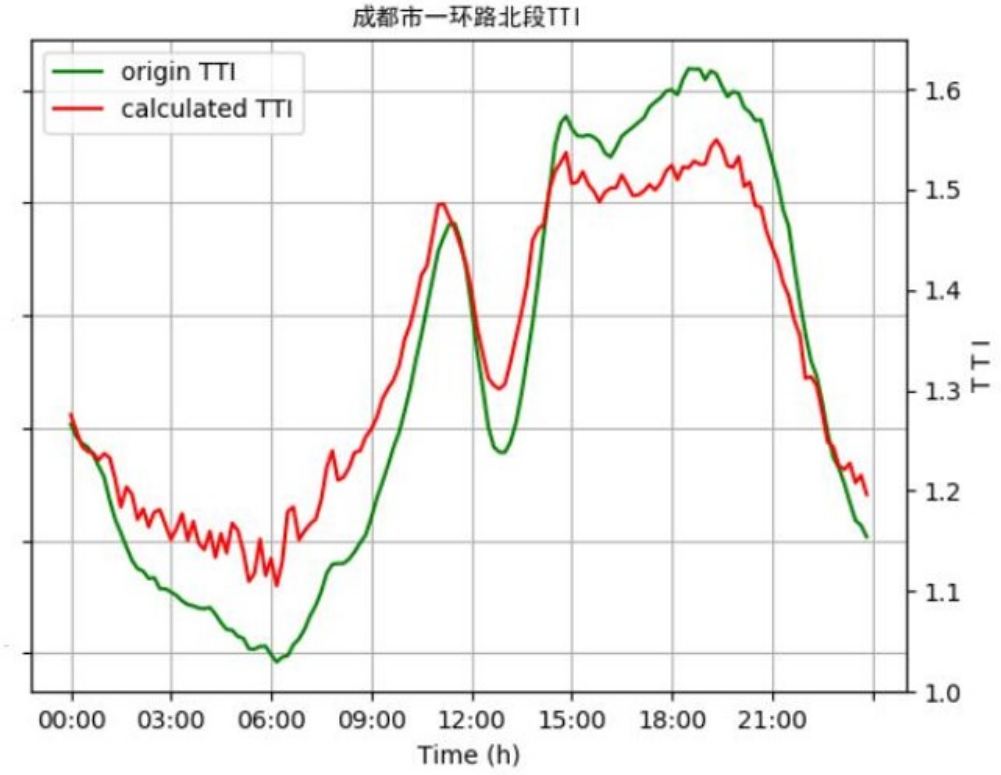
3.3 Result



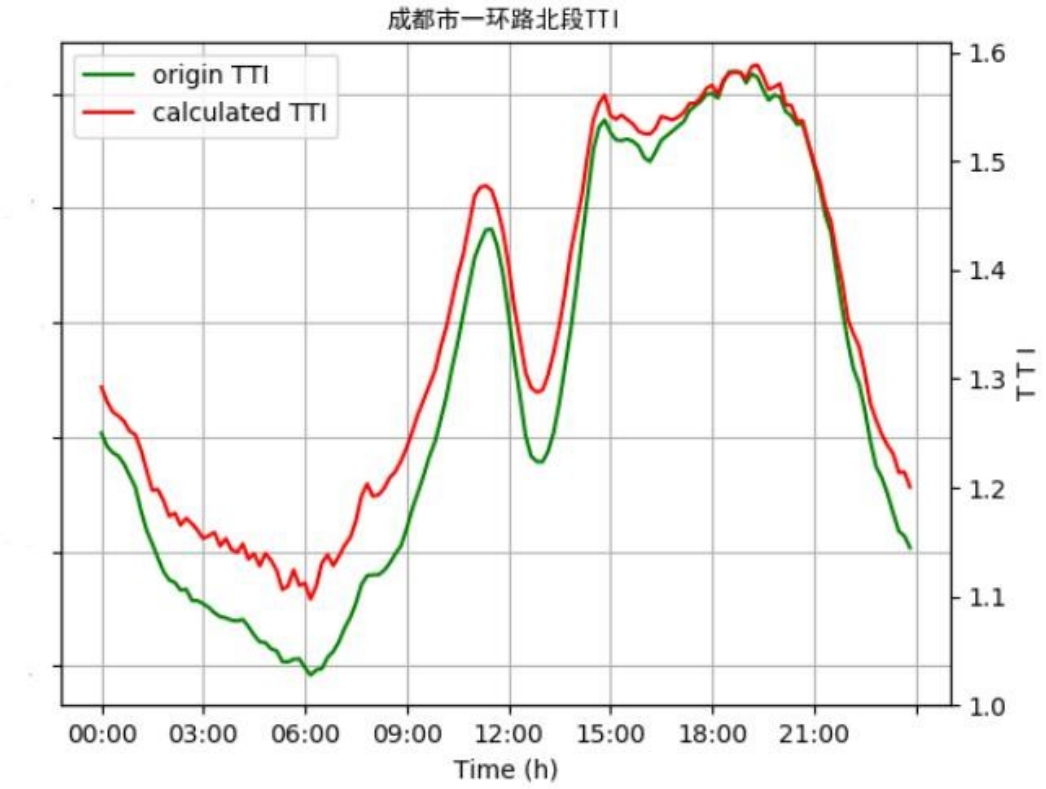
There is a big difference between the calculated TTI and the reference TTI. The most likely reason is that the average speed of the road section used in the calculation is quite different from the official calculation method.

To improve it:

First, when calculating the average speed of a section, the track is no longer the whole track, but only the track that overlaps with the section is used to calculate the speed.



Secondly, considering that the traffic on the road is two-way, and there has no relationship between two directions of traffic flow. When calculating the trajectory velocity, further screening is carried out to select only trajectories in the same direction.



3.4 Error Analysis

Although the calculation model of TTI has been improved, there are still some errors between the obtained results and the official reference values. Possible reasons are as follows:

1. The official TTI calculation adopts the speed of all vehicles, but I only have the trajectory data of taxis while do the calculation.

2. The official may obtain the vehicle speed through the speed camera sampling in multiple areas of a road, and the way to obtain the vehicle speed is quite different.

3. There is no standard answer to the selection of free flow speed. Here, different values may be selected as the free flow speed. And the results can be quite different.

4 Implementation of Supersegment

4.1 Dataset

- Data source: Didi Chuxing GAIA Intiative
- Region: Chengdu
- Time: 2018-10-01 to 2018-12-01
- Content:
  - GPS track of taxis with timestamps
  - Coordinate of road and district boundaries
  - TTI and average speed of districts

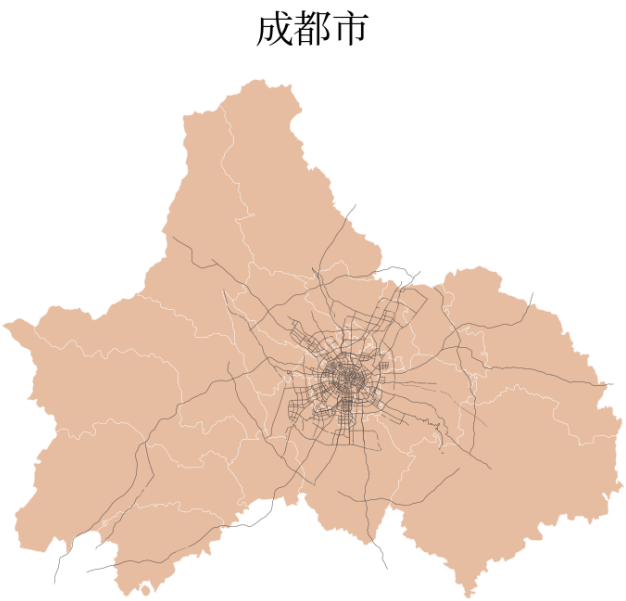


Figure 6: Cheng Du

4.2 Model Design

In this section, I will propose a model to compute Supersegments. At the very first, I need to ensure that I will base on **speed** to do the division. There is a three-step approach of my design:

1. Locate GPS coordinates of taxis into corresponding roads
2. Use the timestamps to calculate average speed and regead it as the instantaneous speed of midpoint
3. Apply clustering algorithm to these midpoints

First, since we know the boundaries of the roads, we can determine the coordinate is on which road. After that, we draw the GPS points on the road. Because we know the timestamp of each point, we can calculate the average speed of two adjacent points and just consider it as the speed of their midpoint. Note that the closer the points are, the closer the average speed is to the actual instantaneous speed. So the best way is to select adjacent points if our data is abundant.

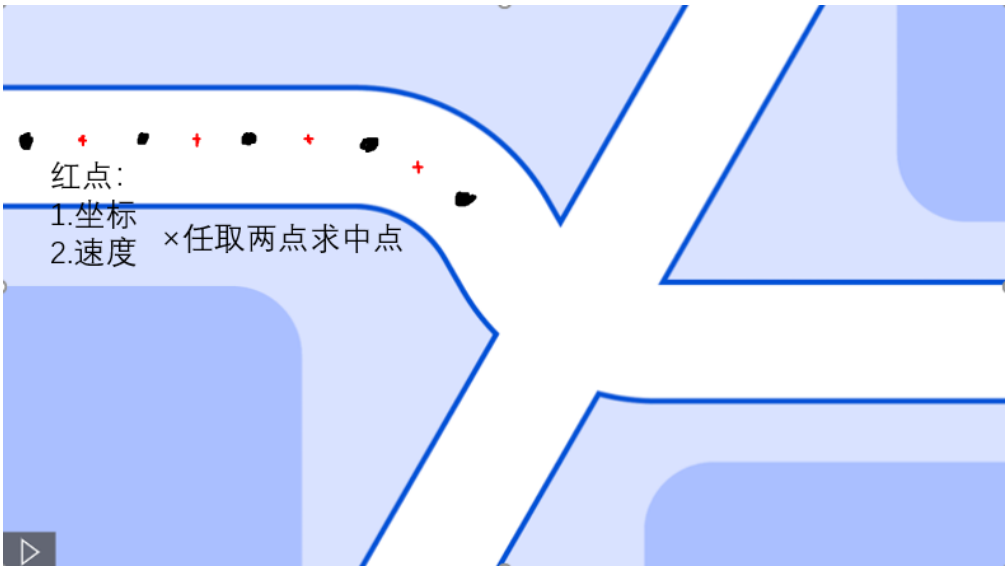


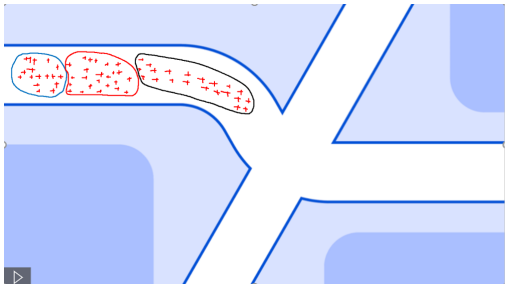
Figure 7: Find the midpoints

Next step is to apply clustering algorithm like *K-Means* to find a certain number of clusters of the midpoints. Every midpoint has three features:

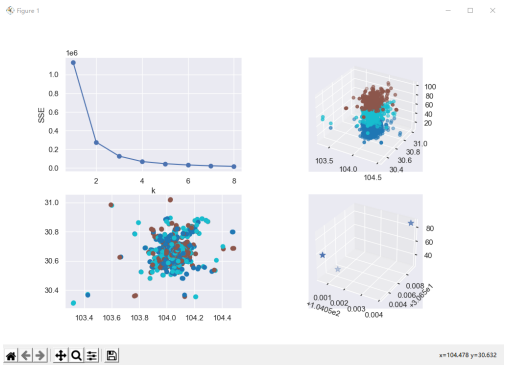
- longitude

- latitude
- speed

So actually this is a three-dimension clustering, but the result we need is two-dimensional, which means we need to balance the weight of these features carefully to get a correct partition. And finally, these clusters give us segments.



(a) Clusters



(b) K-Means Example

4.3 Implementation

Firstly, we show the tracks of taxis on the map:

Figure 8: Tracks

For convenience, we choose a road at the left bottom corner, which is almost horizontal.

The range of the road is about $(104.06060, 30.66680)$ to $(104.07049, 30.66718)$. Then screen out the points.

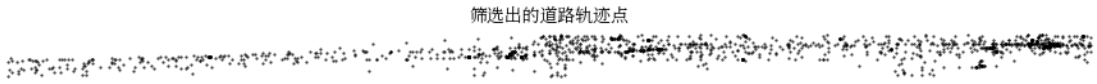


Figure 9: Points

After that, we calculate the coordinate and speed of midpoints. Note that there is no need to compute the actual speed, so the fomula is given below:

$$v = \frac{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{t}$$

where x is longitude, y is latitude and $t = 10s$ is given by the dataset.

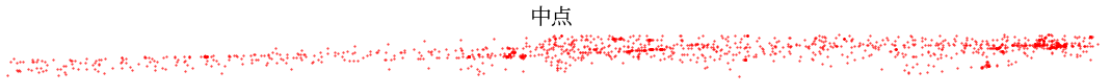


Figure 10: Midpoints

Then apply *K-Means* algorithm on the midpoints.

Using 5000 taxis' data, and determine the value of k is 4.

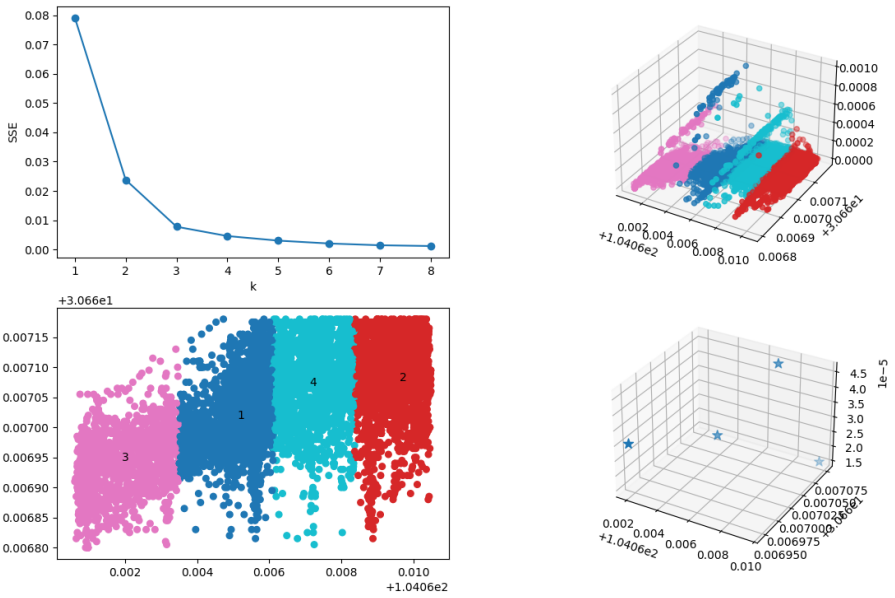


Figure 11: K-Means

Using 10000 taxis' data:

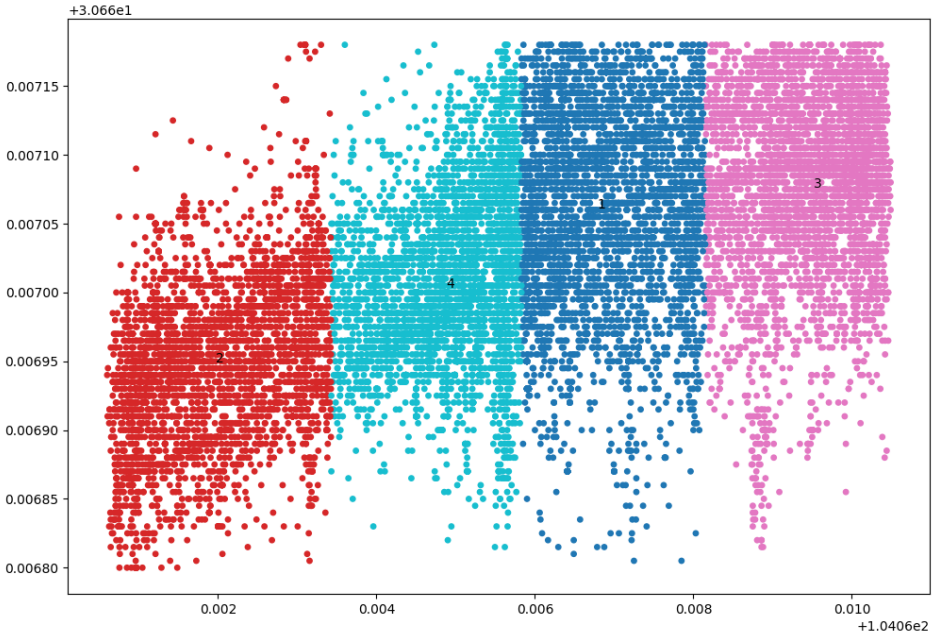


Figure 12: 10000 Taxis K-Means

We already have a fairly acceptable result, but still can be more accurate.

Using 20000 taxis' data:

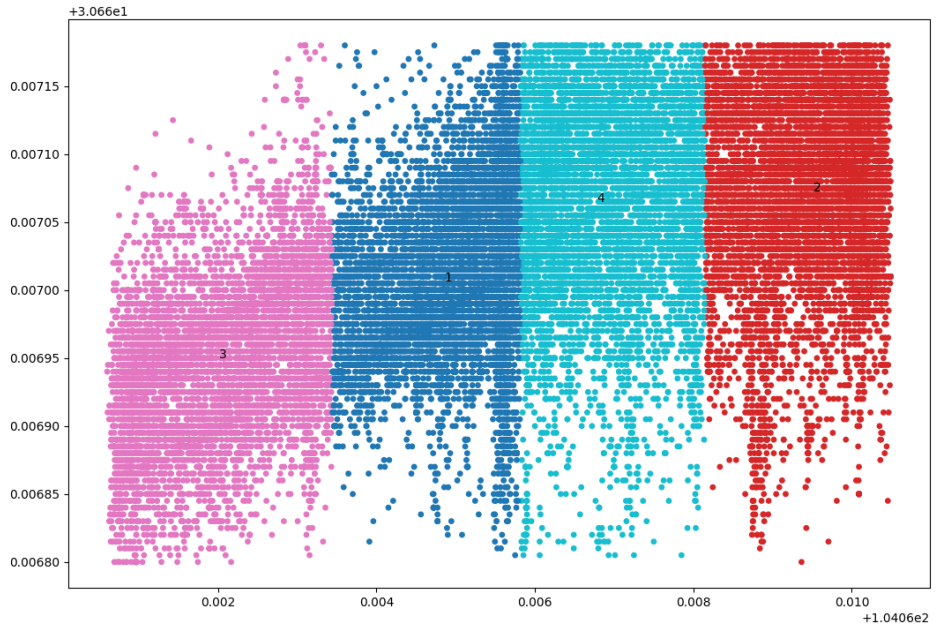


Figure 13: 20000 Taxis K-Means

Thus, we end up with using the data of 20000 taxis, including 36940 GPS points and 36939 midpoints. And we get a really nice result of segments.

4.4 Result

Here we get 4 segments, partitioned by longitude.

1. [104.06060, 104.06346]
2. [104.06347, 104.06580]
3. [104.06581, 104.06814]
4. [104.06815, 104.07049]

And their speeds are $2.92534 \times 10^{-5}, 2.96561 \times 10^{-5}, 3.61342 \times 10^{-5}, 1.63441 \times 10^{-5}$, which meets our expectation of the distribution of speed.

In conclusion, the algorithm we proposed for computing *Supersegment* is practical.