Traffic Network Flow Estimation Based On Social Network Influence Model

First Report

11812804 董　正
11810419 王焕辰
11811305 崔俞崧

Supervisior: 宋轩

Department of Computer Science and Engineering

Oct. 2021

Contents

1 Preliminaries

1.1 Introduction

In this semester, we will try to build a traffic flow estimation system based on graph neural network and social network influence model. Breifly, the structure of the whole system is

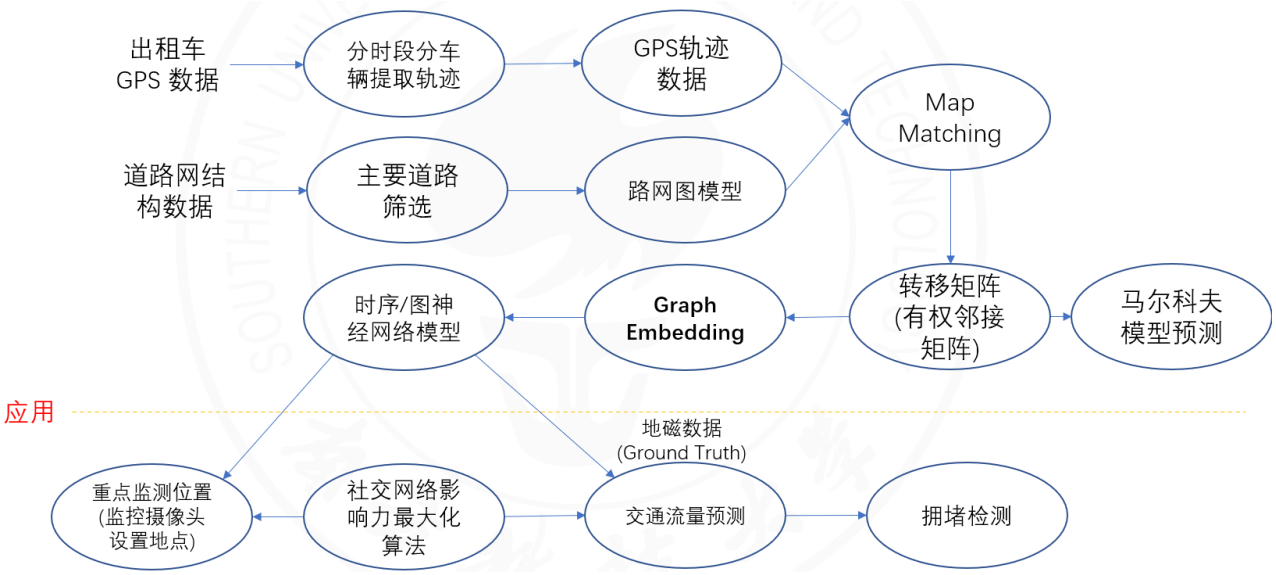


Figure 1: System Structure

1. Process taxi GPS data to get tracks
2. Process road network data to get a basic graph model
3. Match tracks to each road and get the adjacent matrix of the graph
4. Try a simple prediction based on Markov model
5. Graph embedding (the most important part)
6. Build a neural network model for prediction
7. Combine social network influence algorithms to predict traffic network flow, use geomagnetic data as one of the ground truth
8. Applications: traffic surveillance camera position and traffic jam detection

1.2 Report Contents

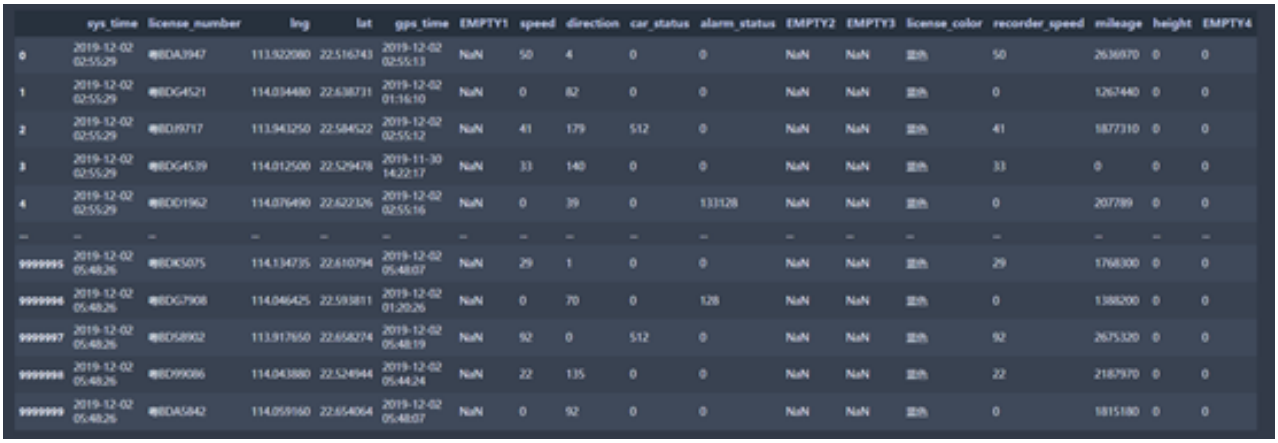
Breifly, we will state our work in this report as

- Taxi Data Processing by 崔俞崧
- Geomagnetic Data Processing by 王焕辰
- Road Graph Model & Map Matching by 董正

2 Taxi Data Processing

2.1 Dataset

- Data source: Shenzhen Municipal Government
- Region: Shenzhen
- Time: 2019-12-01 to 2019-12-13
- Content: Taxi vehicle trajectory data
  - License number
  - Longitude and latitude
  - Speed
  - License type



	sys_time	license_number	lng	lat	gps_time	EMPTY1	speed	direction	car_status	alarm_status	EMPTY2	EMPTY3	license_color	recorder_speed	mileage	height	EMPTY4
0	2019-12-02 02:55:29	粤BDA3947	113.922080	22.516743	2019-12-02 02:55:13	NaN	50	4	0	0	NaN	NaN	蓝色	50	2636970	0	0
1	2019-12-02 02:55:29	粤BDG4521	114.034480	22.638731	2019-12-02 01:16:10	NaN	0	82	0	0	NaN	NaN	蓝色	0	1267440	0	0
2	2019-12-02 02:55:29	粤BDJ9717	113.943250	22.584922	2019-12-02 02:55:12	NaN	41	179	512	0	NaN	NaN	蓝色	41	1877310	0	0
3	2019-12-02 02:55:29	粤BDG4539	114.012500	22.529478	2019-11-30 14:22:17	NaN	33	140	0	0	NaN	NaN	蓝色	33	0	0	0
4	2019-12-02 02:55:29	粤BDD1962	114.076490	22.622326	2019-12-02 02:55:16	NaN	0	39	0	133128	NaN	NaN	蓝色	0	207789	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9999995	2019-12-02 05:48:26	粤BDK5075	114.134735	22.610794	2019-12-02 05:48:07	NaN	29	1	0	0	NaN	NaN	蓝色	29	1768300	0	0
9999996	2019-12-02 05:48:26	粤BDG7908	114.046425	22.593811	2019-12-02 01:20:26	NaN	0	70	0	128	NaN	NaN	蓝色	0	1388200	0	0
9999997	2019-12-02 05:48:26	粤BDS8902	113.917650	22.658274	2019-12-02 05:48:19	NaN	92	0	512	0	NaN	NaN	蓝色	92	2675320	0	0
9999998	2019-12-02 05:48:26	粤BD99086	114.043880	22.524944	2019-12-02 05:44:24	NaN	22	135	0	0	NaN	NaN	蓝色	22	2187970	0	0
9999999	2019-12-02 05:48:26	粤BDA5842	114.059160	22.654064	2019-12-02 05:48:07	NaN	0	92	0	0	NaN	NaN	蓝色	0	1815180	0	0

Figure 2: Dataset

2.2 Data Processing

Because trajectory data are unprocessed original data, there are error data in the process of data acquisition, in addition, the format of original data does not meet our requirements, so the data need to be screened to a certain extent.

1. Sort by GPS time and remove duplicate data of the records.
2. Remove data with the wrong date. There is data loss when recording, and there are many data records outside the specified date range that need to be filtered.
3. Remove the records of speed 0 and speed record too large (>120km/h). The records we needed is the track information generated when the vehicle is running instead of when the vehicle is standing still. The record with the speed 0 has no impact on the track. And the record with excessive speed is obviously the error data. These data need to be removed.

4. Delete the records that cannot be matched with the road segment in the matching process. What is needed in processing is the traffic flow information on each road section, and the records that cannot match with the road section have no use significance.

2.3 Data Visualization

Statistics the speed distribution and draw the speed distribution figure. Then processing the speed data using the statistics result.

We can see that there are vast number of 0 speed records and some records with speed over 150 km/h. These records need to be processed to meet the demand.

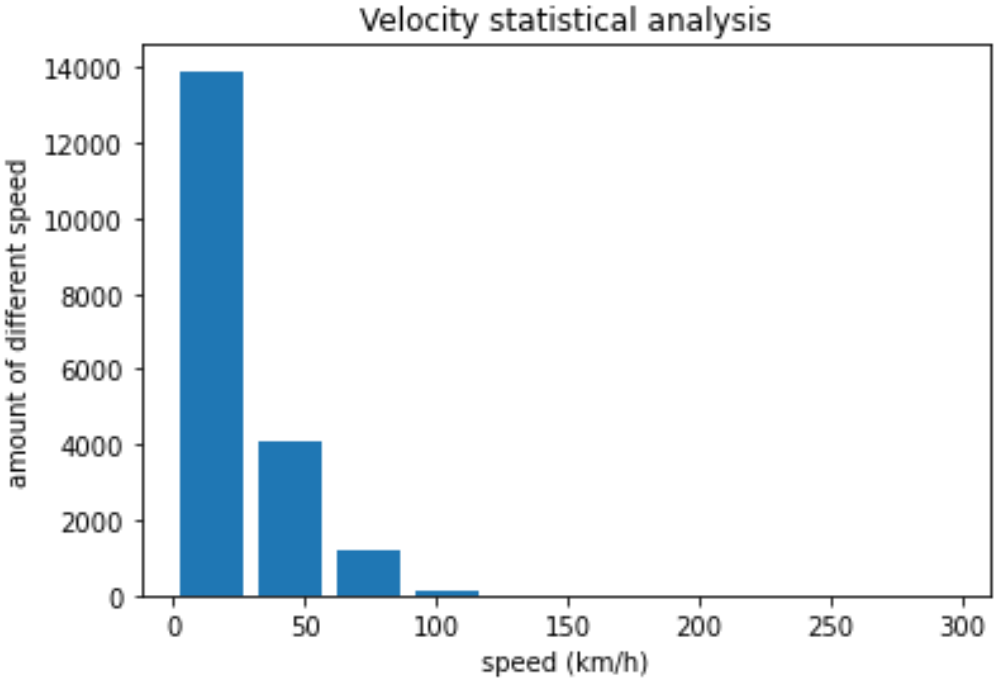


Figure 3: The original speed distribution

After processing, the distribution of speed records are very close to reality. Most of the vehicles records are in urban areas so the speed is not very large, while a small number of vehicles records are on expressways where the speed value is relatively high (about 80 km/h).

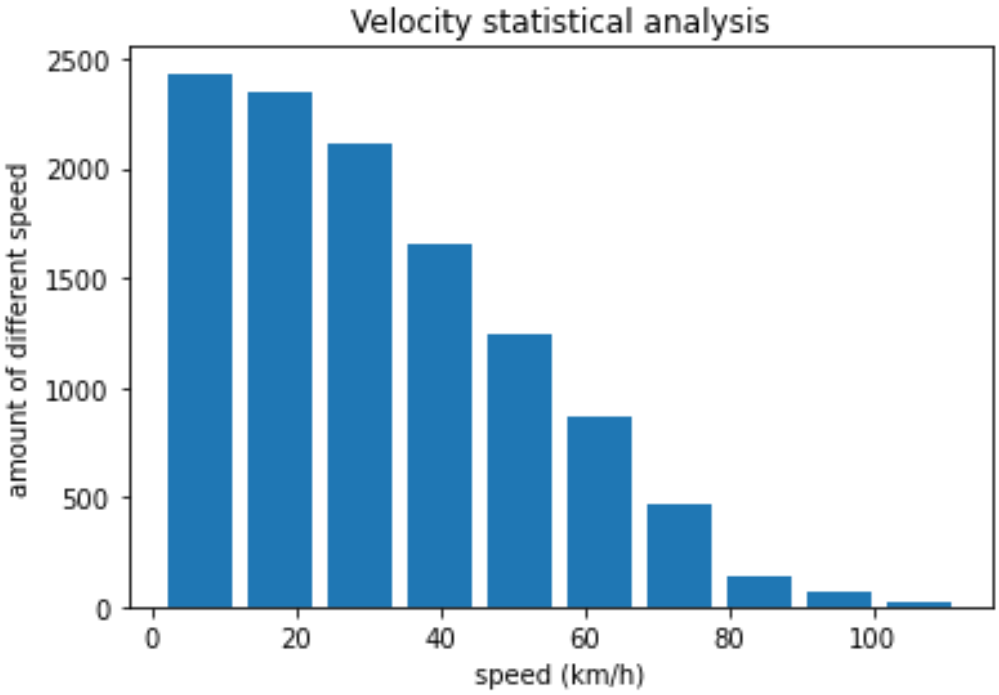
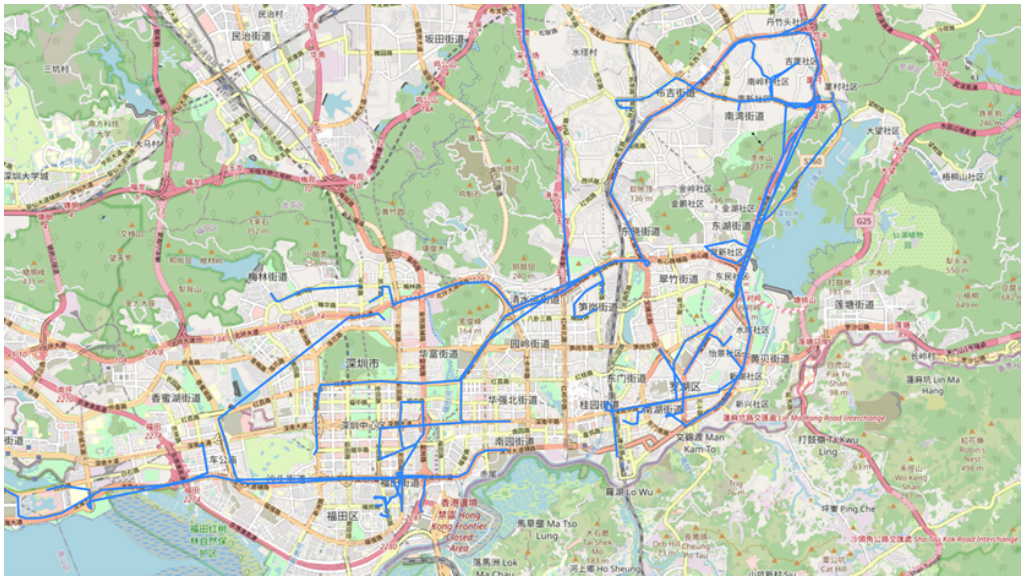
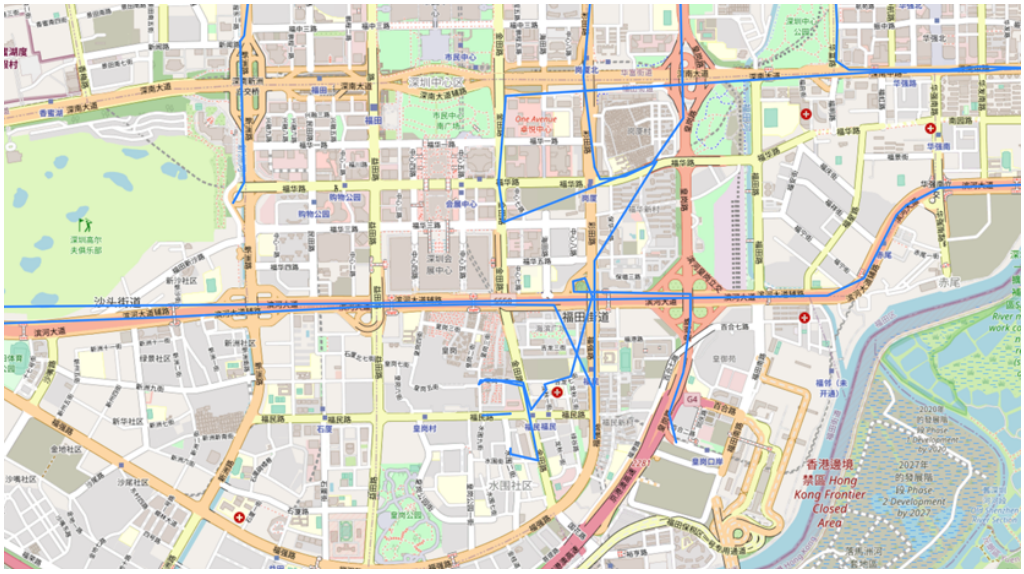


Figure 4: Speed distribution after process

Visualize the all-day trajectory of a vehicle, the results can be obtained as shown in the figure below (Classify taxi tracks according to different orders through GPS time, so as to better distinguish different tracks)





3 Geomagnetic Data Processing

4 Road Graph Model & Map Matching

4.1 Road Network

Use `osmnx` library of *Python* to get the road network graph of Shenzhen. In addition, specify the type as "drive" in order not to get pedestrians and so on.

Figure 5: Road Network

In this graph, node represents the point of intersection of roads. And edge represents road, whose weight contains the geometry information of the road. However, in our model, we want to set roads as nodes, and edges only represent connectivity. Use `networkx.line_graph()` to transform.

The network is still very complex. Therefore, the next step is to choose main roads to simplify the network. In edge weight, we find there is an attribute called `highway`. According to the document of *OpenStreetMap*, we chose some of the types as the major, and filtered all the roads.

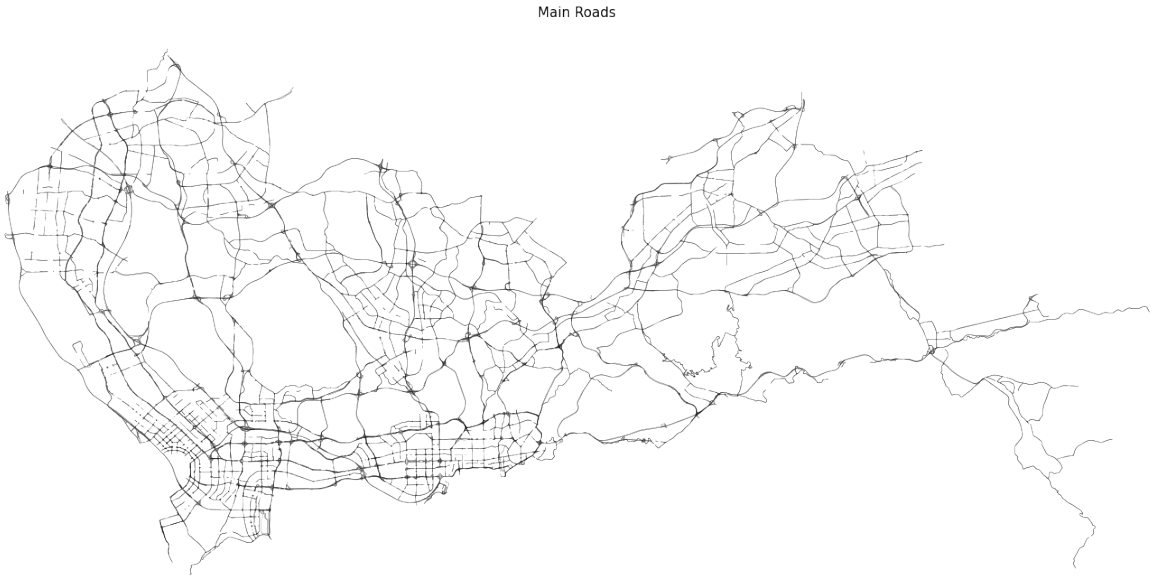


Figure 6: Main Road Network

4.2 Map Matching

In our dataset, a road is represented as a line, however, it should be an area in real world. Besides, it is hard to match a point to a 2-D line. Therefore, we need to convert a road to an area in advance.

Use method `buffer()` in *shapely* package.

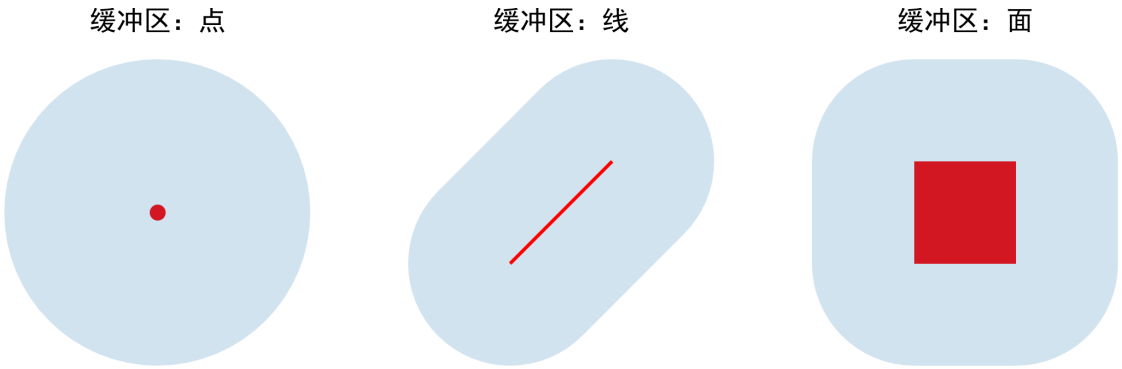


Figure 7: `buffer()`

The `buffer()` method will convert a `Line` to a `Polygon`.

After that, we take the advantage of the continuity of tracks to do map-matching.

1. Rename the id of every remaining road

2. Initialize transition matrix
3. Downsample the track, delete the points which are very close in time (<30s)
4. Find the center of every road
5. Find the median of the track
6. Sort all the roads according to the distance between the center and the median
7. For each GPS point, use `contains()` function to match it to a road
8. Modify transition matrix

The expected time complexity is about $O(n^2 \log n + Cn^2)$.

Finally the output is a weighted adjacent matrix (transition matrix), which is the true representaion of the graph.

After that, we can do a simple statistical prediction.

What's more,

- Split tracks to different time intervals to increase the accuracy of prediction
- Use multiple processers to accelerate