Tongle Yao

Advisor: Professor Desheng Zhang

Independent Study

16th December 2018

Research Report

1. independent Study Report

In the Semester of Fall 2018, I had a chance to participate the independent study with Professor Desheng Zhang and his student Yu Yang. The objective of this study including (i) how to collect city road and boundary from open source (ii) analyze, process and format traffic data (iii) visualize road structures and the traffic density on the map. At the beginning of the semester, I plotted 30 cities' highways and recorded their information by using OSMnx and QGIS. Then, I analyzed car's data nearby Shenzhen's toll stations. The last part of the study is to highlight millions of car data in several cities on the map for analyzing mobility patterns of cities. The achievement of these works can help people easily understand a city's mobility pattern.

I started this project totally as a rookie who knows nothing about visualization, processing data, even python. When I was doing these projects, I encountered a lot of challenges, such as filtering highway from osm, projecting road and city boundary properly, mapping car coordinates to road or toll station and coloring trajectories of car on the map beautifully. Although it took me a lot of time to solve all these challenges, I learn a lot by solving them and grow up from a rookie to someone who can produce nice data visualization. I am going to show some problems I encountered during the study and the final achievement below.

2. Problems and solutions

2.1.1 Road data acquisition from OSMnx

Acquiring road information from OSMnx is not an easy process. I need to collect information from 32 cities and it is different to return cities road information directly from

OSMnx. I found that OSMnx returns pointer in some cities but returns polygons in others. I can only query city's information from OSMnx by using polygons. Thus, Pointer is going to result in exception when I try to get the city's road information (shown in Figure 1). After carefully reading their API and OSMnx's author Geoff's answer online, I learn that sometimes OSMnx put the polygons at the second place. So, I catch the exception and add some extra parameters if current city has the same problem.

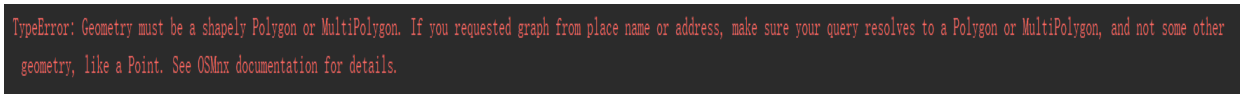


Figure 1

Another problem of OSMnx happens when we try to get id of the start node and end node of each edge. OSM splits each road into several edges and assigns ids to both ends of the edge for querying information including length, type, etc. Some small edges in the road are missing and do not have node information. After reading some author's article, I noticed that network simplification is done by OSMnx automatically under the hood, which means it does not contain all the nodes on OSM. Yu Yang and I discussed several times about this problem and try to find out if there is a way to get the end points id through edge id. We failed in the end because osmnx does not support such a query. Fortunately, we do not need so many nodes for now. But unfortunately, it caused a even bigger problem later. I will talk about it in 2.4.2.

2.1.2 achievement of road data acquisition from OSMnx

Finally, at this part, I fill a form about 32 cities basic road information including length of highway, number of nodes, etc. and use file generated by OSMnx to produce city road maps in QGIS. At this point, I get familiar with python and OSMnx. Also, I learned how professions such as OSM deal with map data and started to conduct some simple visualize applications.

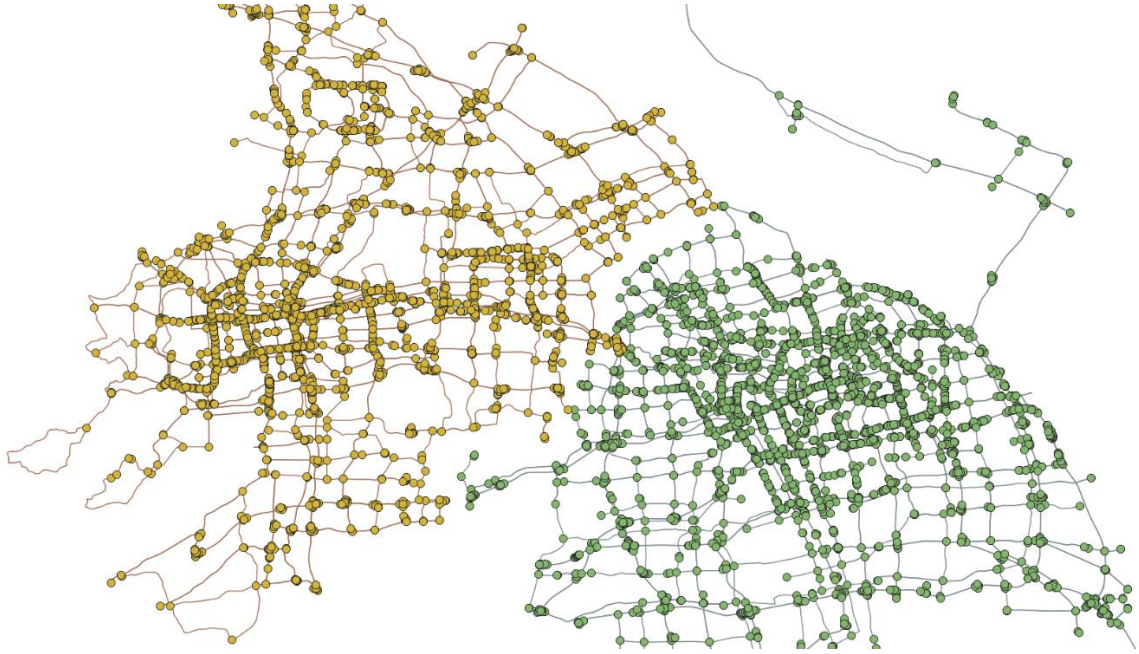


Figure 2

Figure 2 is a simple demo of shanghai and suzhou. I aquire shapfile through OSMnx and show it with QGIS. I have another csv file with all those nodes and edges information like GPS coordinates, length and type.

2.3.1 purpose of analyzing Shenzhen toll gate

I started to analyze data related to Shenzhen toll gate after getting familiar with processing data. The goal of this part is to map cars into the nearest toll gate base on their coordinates and analyze the average, median and standard deviation of cars' speed in different time intervals. I finally find the speed deviation of cars in each gate compared to the average speed.

2.3.2 Problem of analyzing Shenzhen toll gate

Here I met a problem when I tried to find the nearest gate of each car, how to calculate the distance between two coordinates, cars and toll gates. At the very beginning, I thought it should be very easy, I just mathematically calculate longitude and latitude and I can get what I want. However, my code results in bad consequence that it maps all cars in the wrong gate. It seems like I mess up the equation because I simply think the surface of earth is flat which is definitely wrong.

Yu Yang helps me out at this part. He told me I can use his code to calculate distance between two coordinates. After using his code, I successful map cars into the nearest toll gate. I learned the relationship between coordinates, distance and the projection on earth. This helps

me a lot when I do the final project.

2.3.3 enlighten of analyzing Shenzhen toll gate

station	type	0	1	2	3	4	5	6	7	8	9	10	11	12
0	avg	16.59223	19.08571	20.6	12.77778	19.08696	19.53333	19.83221	15.34956	15.1875	16.94199	15.49784	24.7929	10.43846
0	median	16	19	21.5	14	19	21.5	21	15	15	17	16	20	9
0	avg_in_ran	16.59223	17.22464	17.6519	17.38922	17.59474	17.85909	18.65583	16.83557	16.28904	16.43997	16.31886	17.0473	16.62416
1	avg	13.36842	0	0	0	0	21.75	16.10476	14.73819	14.05328	15.76207	15.60741	14.03636	8.057692
1	median	13	0	0	0	0	24	16	14	12	16	14	11.5	5
1	avg_in_ran	13.36842	0	0	0	0	15.85185	16.01887	14.94887	14.62804	14.90301	14.97446	14.79054	13.70609
2	avg	30.64706	0	0	0	0	0	22.75	22.51563	24.02	14.90541	19.24793	23.39806	24.13433
2	median	32	0	0	0	0	0	19.5	20	24.5	13	17	14	23
2	avg_in_ran	30.64706	0	0	0	0	0	29.14286	23.44966	23.59296	21.2381	20.6269	21.20121	21.54965
3	avg	52.64286	20.67442	0	0	0	15.40741	15.44444	23.92994	21.86441	17.83945	28.89189	26.87671	21.31148
3	median	58.5	21	0	0	0	10	18	26	23	16	32	26	22
3	avg_in_ran	52.64286	28.52632	0	0	0	24.30952	23.45161	23.752	22.96956	21.23566	22.02364	23.15672	23.04404
4	avg	0	37.7	48	0	0	34.25	0	43.35897	38.38889	21.30769	38.29412	41	32.25
4	median	0	39	50	0	0	33	0	45	47	20	41	46	31.5
4	avg_in_ran	0	37.7	41.35484	0	0	40.54286	0	42.02703	40.83636	38.77236	38.66879	39.42381	38.57983
5	avg	0	28.61	31.63636	0	0	31.07692	0	24.12	25.7418	20.27757	15.3487	23.64078	22.29524
5	median	0	29	34	0	0	31	0	27	27	21	13	24	23
5	avg_in_ran	0	28.61	28.90991	0	0	29.32117	0	25.66234	25.6898	24.22085	21.83486	22.17868	22.18576
6	avg	19.88889	30.93333	55.6	0	0	0	0	32.65217	21.80791	18.51095	17.79524	23.03955	16.84211
6	median	20	28	56	0	0	0	0	30	19	19	13	18	12
6	avg_in_ran	19.88889	27.77778	31.58904	0	0	0	0	32.23936	27.18082	24.81474	22.54379	22.64294	21.51333
7	avg	26.36667	52.5	31.55	0	0	0	3.142857	17.59391	17.19014	17.9382	16.455	34.55172	17.45545
7	median	27	52	32	0	0	0	3	13	13	17	15	42	14
7	avg_in_ran	26.36667	32.9	32.45	0	0	0	29.38806	20.20307	19.14144	18.77281	18.09499	18.68564	18.54895

Figure3

Shenzhen toll gate analyzing project is the first time I map a car coordinates to the nearest object. It provides me valuable experiences for the next project. Also, I practice reading and writing csv in python through this project. The achievement of this project can help people analyze cars' mobility modes near toll gates. People can see when toll gates has more traffic or car drive fastest at all the time periods.

2.4.1 Trajectory visualization

This project is to demonstrate traffic situations of a city in some specific moments. The first part of the project is to process data. I got hundred thousands of roads and millions of traffic data. This part asked me to map these traffic data into the nearest road, just like what I did in the ShenZhen toll gate project. After I got all the road with how many cars drive on that road at a time point, I was able to visualize the traffic situation with different colors. Although former project helped me gather some experiences, I still met tons of problem in this project.

2.4.2 Problem in Trajectory visualization

The first problem I met is efficiency and hardware limitation. The input data is so huge, and it is not possible to run this project with a brute-force approach, which results millions of comparisons. After talking to Yu Yang, I decided to hash the city into small blocks and put road into these blocks first. Then, it uses the same function to find the block that traffic coordinate falls into. As long as I hash the map into blocks small enough, the number of comparisons can

be reduced to a acceptable number. Since the boundary file has parameter that shows city as rectangular, dividing city into small blocks is not that hard. However, it is so difficult to figure out those geometry part.

I was lucky that Yu Yang has provided the function to convert geographic to web Mercator. So I do not have to deal with the complex coordinate system.

Another problem I met when I process data is memory limitation. Since I need to add a lot of attributes on the rode such as endpoints coordinates, length, weights, I create an object for each road. When I tried to run the code on my laptop, the memory explodes because it cannot hold hundred thousands of objects in the memory. After my mentor's reminder, I switch to use dictionary instead of object. Also, my new desktop helped me solve this problem.

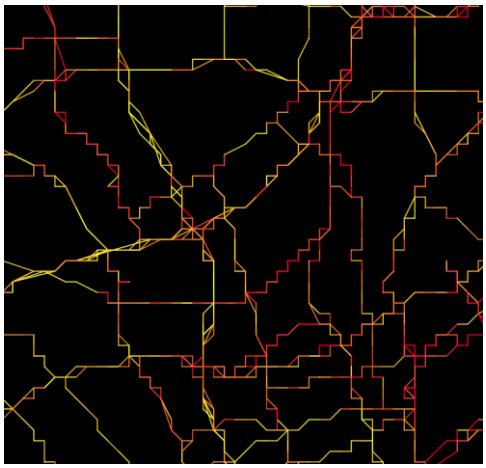
After overcoming these difficulties, I got a csv with processed data. It is the time to visualize these data. I choose processing to visualize my data. It is a great light software but its interface and module is not so friendly. I met a lot of weird problem because the Python module is actually driven by Java. Luckily, none of these problems bother me too long.

(before) (after)

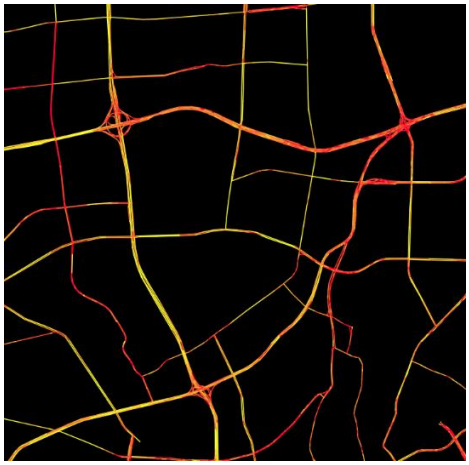
For the plotting part, the first problem I encountered is a ratio problem. I figure out how to draw on processing by reading the demo Yu Yang provides me. I did not realize this problem since the sample project does not need the ratio. After running my code, roads on the image look skew and does not match city's boundary. Originally, I thought it is projection problem such as convert coordinate in the wrong way. So, I went to Yu Yang, but the projection looks

fine. He mentioned that this is probably we pass wrong ratio of size into the projection. I calculate the ratio when the code reads the city boundary and set the parameter properly. Then, the map looks much normal.

The second problem I encountered in the plotting part is the hardest one I think. After magnifying the image, I noticed something goes wrong with the edge. All the turnings look like a right angle. In the beginning, I thought it is because of smooth setting in processing. I tried to set smooth function in processing, but it did not work. Then, after discussing with Yu Yang, we thought it is probably because OSMnx's simplification results missing edges. Then, we used edges in Yu Yang's database instead of OSMnx's data I got in the first project. Theoretically, Yu Yang's data contains much more edges which can make the rode looks smoother. However, nothing has changes. I took lots of time to figure out how processing works with the map. Accidentally, I find an answer on stackoverflow mentioned processing has its own map function while I was using the map function from the demo project. So, I just used the processing map function to replace the previous one and I got the smooth, beautiful line.



(before)



(after)

The last problem is how to color the edge. I got the weight of each node when I processed the data. At first, I just linearly change the weight of green in RGB to change the color. However, most edge only contains dozens of cars which is not enough to mark the difference of weight between each edge.

Then, I tried to squire the weight to make some difference. I got the figure below which has more yellow edge. It looks better but I was still not satisfied since it looks monotonous. Due to square root of the 255 is 15.9, every edge contains more than 15.9 or near 15.9 are yellow. Thus, there are only yellow and red in the graph and all the colors between them are missing.

So, I went to another function with flatter curve, log. The helps me create a graph with yellow, red and some orange edge. This graph perfectly shows the traffic situation of the city.

2.4.3 Achievement of Trajectory visualization

In this project, for the first time, I deal with big data, mapping coordinates and visualize it on graph with beautiful lines. I learned how to optimize algorims to reduce the amount of calculation, geometry graphing, using math function to solve problem. By using my code to visualize more city trajectories in different time can, it helps people understand the mode of urban traffic.

3. Summary

Throughout the semester, I have done a lot of data processing, analyzing, and visualization work, which helped me learn a way to demonstrate data to people. I start to understand why we have to analyze data, the significance of visualizing the data and how to do it. During the learning process, I have the chance to get familiar with lots of new softwares such as python, processing, OSMnx. I was a totally rookie of python at the beginning of the semester. Now, I can use python to process, analyze and output tons of data. Other course can provide me some basic program training, but this course gives me a window to see how people program in the real world. All the knowledge I have learned in this course will help me easier integrate with the company in the real word.

4. Conclusion

Finally, I would like to give great gratitude to Professor Zhang for giving me this opportunity to participate in this fantastic project. Also, I must thank Yu Yang for designing such a great learning plan to guide me to finish the final project. This course gives me a great insight of the real-world problem. What I learned from this course will help me solve problems no matter in school or in the work.