Data Wrangling with MongoDB on OpenStreetMaps

1. Map Area

Singapore

I have selected Singapore as it is the place I live and work. Also, I am working on my PhD Thesis that requires me to pull some data from OSM so it is useful to work on Singapore data. The file I downloaded is 379.8 MB.

2. Procedure

First, I do the counting of the tags according to the first exercise in the nanodegree, using mapparser.py.

It gives me this; {'bounds': 1, 'member': 130673, 'nd': 2087568, 'node': 1667063, 'osm': 1, 'relation': 2974, 'tag': 870923, 'way': 264907}

The bounds for this file is; bounds minlat="0.807" minlon="103.062" maxlat="1.823" maxlon="104.545" which is larger than Singapore so I need to get rid of places that is not in Singapore first. But I leave that to next stage after I pass the data to database.

2nd step in working in the data is to check the tags in the file. My analysis using tags.py gave me this result. {'lower': 681314, 'lower_colon': 176410, 'other': 13196, 'problemchars': 3}

3rd step is to check the number of users that worked in this part of the OSM map. And running users.py gives me all the unique users that worked in this file, which is 2270.

In the auditing phase, 2 things I have noted first one is since I have data not only from Singapore but also Johor Bahru, which is the neighbouring city, in Malaysia, I need to check if the way is in Singapore or in Malaysia.

And secondly the main assumption used in the class that In English street names end with street type does not apply here since there are 4 official languages in Singapore and for example a street name can be Jalan Bukit Kakut, Jalan in this case is street for malay and it is in the beginning of the street name not in the end. For more information check here;
https://en.m.wikipedia.org/wiki/Road_names_in_Singapore

So I needed to do the audit twice first for english by checking the ends of the words. I used audit_street.py for this purpose. The results are in the streets.txt. And I did the same for malay words which are in the beginning with audit_street_malay.py.

Lastly by using data.py, I converted the osm XML file into json file which will be used in the mongodb database. During the conversion to json, I corrected the street names and deleted postcodes that is less than 5 digit. Then I created a database named osm and collection named Singapore.

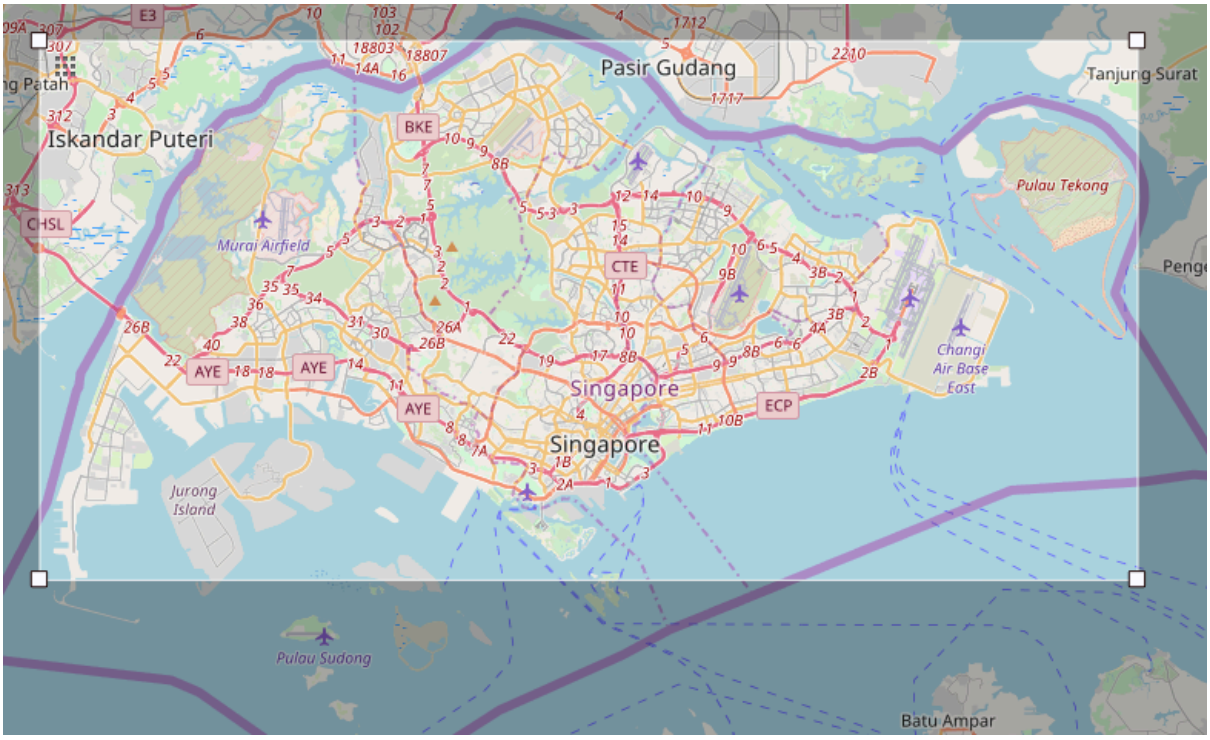
3. Data Overview

Now that my data is in mongoDB, at this stage I can do queries on the database to reveal more information about the data.

Let's find out how many nodes are inside my lat-long.

```
> db.singapore.find({pos: {$geoWithin: {$box :
[[0.807, 103.062], [1.823, 104.545]]}}}).count()
1667063
```

But Singapore is actually smaller than this area, so let's check it this area which is; 1.23, 103.6 & 1.47, 104.09.




```
> db.singapore.find({pos: {$geoWithin: {$box :
[[1.23, 103.6], [1.47, 104.09]]}}}).count()
1215448
```

Count of the amenities in the area;

```
db.singapore.aggregate([{$group: {_id:"$amenity", count:{$sum:
1}}}, {$sort:{count: -1}}])
```

```
{ "_id" : null, "count" : 1918406 }
{ "_id" : "restaurant", "count" : 2421 }
{ "_id" : "parking", "count" : 2247 }
{ "_id" : "place_of_worship", "count" : 1084 }
{ "_id" : "atm", "count" : 837 }
{ "_id" : "school", "count" : 775 }
{ "_id" : "cafe", "count" : 596 }
{ "_id" : "fast_food", "count" : 451 }
{ "_id" : "fuel", "count" : 426 }
{ "_id" : "bank", "count" : 354 }
{ "_id" : "taxi", "count" : 338 }
{ "_id" : "shelter", "count" : 323 }
{ "_id" : "toilets", "count" : 294 }
```

Count of the streets which are tagged with the tag of “highways” are listed down. Note that residential and service type of streets are more than primary or secondary streets which are bigger in hierarchy.
https://wiki.openstreetmap.org/wiki/Key:highway

```
db.singapore.aggregate( [{$match: {type: "way"}}, {$group:
{_id:"$highway", count:{$sum: 1}}}, {$match: {count: {$gte:
1000}}}, {$sort: {count: -1}}] )
{ "_id" : null, "count" : 148098 }
{ "_id" : "residential", "count" : 39534 }
{ "_id" : "service", "count" : 30801 }
{ "_id" : "footway", "count" : 12015 }
{ "_id" : "primary", "count" : 7217 }
{ "_id" : "secondary", "count" : 5231 }
{ "_id" : "track", "count" : 3509 }
{ "_id" : "tertiary", "count" : 3048 }
{ "_id" : "unclassified", "count" : 2597 }
{ "_id" : "motorway_link", "count" : 1646 }
{ "_id" : "steps", "count" : 1600 }
{ "_id" : "primary_link", "count" : 1534 }
{ "_id" : "trunk", "count" : 1525 }
{ "_id" : "path", "count" : 1413 }
{ "_id" : "motorway", "count" : 1241 }
```

To see which cuisine is more preferred I also checked the cuisine numbers in the area;

```
db.singapore.aggregate([{$match: {amenity: {$exists: 1},
amenity:"restaurant"}}, {$group: {_id:"$cuisine", count:
{$sum:1}}}, {$match: {_id: {$ne: null}}}, {$sort: {count: -
1}}])
{ "_id" : "chinese", "count" : 172 }
{ "_id" : "japanese", "count" : 82 }
```

```
{ "_id" : "pizza", "count" : 52 }
{ "_id" : "indian", "count" : 50 }
{ "_id" : "korean", "count" : 48 }
```

4. Problems

Data wrangling process was not always straight forward. The data might be missing, corrupted or misedited. Common occurrences happen in the street names, postcodes or phone numbers. In street naming, people can use abbreviations that make sense to them, but it might not be same convention, throughout the all data source. Since Openstreetmap is a crowdsourced data, we can see many examples of it. This kind of inconsistent data might be hard to correct. In my case, it was more difficult as in Singapore, street names can be in English or in Malay. Malay street names start with the type of the street first and then the name, ex. Jalan (street in Malay) Melaka, whereas in English it is Melaka Street. So, I needed to check and correct the abbreviations in two steps.

Another possible data inconvenience happens with postcodes. There is no easy way to check the correctness of a postcode. So, I have not checked if a postcode is correctly entered or not. We can be only sure of the postcodes that are missing digits as erroneous and get rid of them.

5. Conclusion

This study is a very good example of data wrangling. We have a crowdsourced data, which is Openstreetmap map data for the area of Singapore. As this is an XML data it needs a procedure to be done from xml to database for this case mongoDB. Although this process can get tedious, the final result is quite helpful.

With this data I am planning to use this in my Phd study which is about the movement of people within the city. Openstreetmap gives me an easy access to the street data of Singapore.

Since I am actually using Openstreetmap data, I did parsing of the data with many tools out of this courses context. I parsed the data for MongoDB and mysql for this course, and POSTGRESQL for my own purpose. I can say the process of parsing data is easier for MongoDB since it uses json. But for me MongoDB queries are more complicated then SQL queries, it might be because I know SQL better. But even though I prefer SQL, I used MongoDb for this course, because I wanted to use this opportunity to learn. And last POSTGRESQL with its POSTGIS extension gives me the capability of plotting the data in a GIS tool which is very useful. I can built my own maps using this data.