OpenStreetMap Data Wrangling with MongoDB

Prasanna Velisetti

Map Area: Singapore

Problems Encountered in the map

Auditing & cleaning the postal codes & phone numbers

Singapore postal codes are always 6 digit numbers where first two numbers denote sector code and rest specific to the delivery point within the sector.

The .osm file is audited for phone numbers and postal codes through audit.py. There is very less data for street names (12) and hence there is no need to clean the street names as we are not going to use them.

Phone numbers are observed to in different formats. Some phone numbers which are not Singapore numbers are being noticed which are toll free or US numbers. So there is no need to clean the data for phone numbers.

As Singapore postal codes are 6 digits, auditing is being done to check the number of digits. There are few postal codes which are 2,3,4 or 5 digits and two postal codes with right number of digits but given in different format(Eg.: 437 437 , Singapore 408564).

The initial .osm file that is extracted from OpenStreetMap is not organized properly for our use and hence it is processed through process.py python code. For cleaning postal codes, the postal codes which are in different format are corrected to be in right format and the postal code with different number of digits than expected are not included in the data. Also as first two digits of postal codes indicate the area in which the place belongs, and last four digits are just the building number, only first two digits are used in the postcodes for analysis purposes.

By running the following query, we will come to know about how many documents have postal codes and how many do not.

```
db.Singapore_Final.count({"address.postcode":{"$exists":1}})
```

No postalcodes : 594212
No addresses : 584006
With postalcodes : 4793

So there are very less documents that have postal codes compared to the total documents but it is good enough quantity for analysis purposes depending on what we need to investigate.

AUDITING STREET NAMES

There is very less count of the documents which have street names in their addresses.

The following query returned 12 documents.

```
db.Singapore_Final.count({"address.street":{"$exists":1}})
```

AUDITING AMENITIES

Among the documents that do not have addresses, if we query to see what are the amenities in these places, most of the places do not have any amenities. The amenities which do not have addresses in the documents are mostly, the parking-lots which makes sense and the second largest count amenity which do not have addresses are the restaurants.

```
db.Singapore_Final.aggregate([{"$match":{"address":{"$exists":0},"amenity":{"$exists":1}}},
                              {"$group":{"_id":"$amenity","count":{"$sum":1}}},
                              {"$sort":{"count":-1}}])
```

Overview of the data

#File Sizes

Singapore.osm 107MB

Singapore.osm.json 127MB

#Number of documents

```
db.Singapore_Final.count()
```

```
599006
```

#Number of nodes

```
db.Singapore_Final.find({"type":"node"}).count()
```

```
510768
```

#Number of ways

```
db.Singapore_Final.find({"type":"way"}).count()
```

```
88230
```

#Area with highest number of amenities

```
db.Singapore_Final.aggregate([{"$match":{"address.postcode":{"$exists":1},"amenity":{"$exists":1}}
},{"$group":{"_id":"$address.postcode","count":{"$sum":1}}},{"$sort":{"count":-1}}])
```

This query returned 06 postal code which is near Raffles Place.

Additional Ideas

IMPROVING DATA

I have seen in the data that there is no specific format for phone numbers or validation for postal codes. Given below are the different formats (around 10) being used in the phone numbers data which are audited through audit.py file.

```
re.compile("([+]\d{2}[-\s]\d{4}\d{4})|([+]\d{2}\d{4}\d{4})|([+]\d{2}[-
\s]\d{4}[\s]\d{4})|(\d{4}\d{4})|(\d{4}[-\s]\d{4})|([+]\d{2}[-\s]\d{2}[-\s]\d{3}[-
\s]\d{3})|([+]\d{2}[-\s]\d{5}[-\s]\d{3})|([+]\d{2}[-\s]\d{4}[-\s]\d{3}[-\s]\d{4})|(\d{4}[-
\s]\d{3}[-\s]\d{4})")
```

We had to process it further to see the right or formatted data. It is not possible to validate all the data but some level of validation can be applied to fields like phone numbers and postal codes. Depending on the country we select, the postal codes are specific number of digits and the same with the phone numbers. It would avoid unnecessary processing time if we can adopt this kind of validation at the time of input of data by the user through implementation of Javascript.

Benefits:

- Saves time during processing data
- More accurate data

Anticipated Problems:

- Validations specific to every country may be difficult to implement

SCOPE FOR FUTURE PROSPECTS

Using this data, we can actually get the list of places which are densely populated with schools or restaurants or commercial buildings or residential areas. Using this list of places, we can look for prospects of a new school or any other amenity.

A sample query is shown below for extracting most densely populated area with schools:

```
db.Singapore_Final.aggregate([{"$match":{"address.postcode":{"$exists":1},"amenity":{"$eq":"scho
ol"}}},{"$group":{"_id":"$address.postcode","count":{"$sum":1}}},{"$sort":{"count":-
1}},{"$limit":1}])
```

So the densely populated area with the school according to the data extracted is Tampines and Pasir Ris.

LIST OF WEBSITES FOR RESEARCH

There is also scope for research on a group of amenities by getting the list of websites and going through them.

```
db.Singapore_Final.find({"website":{"$exists":1}},{"website":1,"_id":0,"name":1,"phone":1})
```

The above query gives a list of amenities, their websites and phone numbers.

Conclusion

From the basic querying, we have seen that a lot of useful information can be extracted and used for our analysis depending on what is our ultimate goal. Given that the information may not be accurate, it however gives a firsthand overview on the data which can be refined later for extended research.