P3: Wrangle OpenStreetMap Data

Valmik Patel

Map Area: Mumbai Metropolitan Area, India

https://www.openstreetmap.org/node/16173235#map=11/18.9527/72.8325
https://s3.amazonaws.com/metro-extracts.mapzen.com/mumbai_india.osm.bz2

1. Problems Encountered in the Map

I have analysed and wrangled 5 data points for this project which are Postal Codes, Street Address, City, State and Country. I will go through each data point, list the problems that I encountered with each of them and how I solved those problems.

Street Address

I used the same methodology used in lesson 6 to find out the street names that do not match the expected. I encountered the following types of problems.

- Indian street names that were not part of the original expected list. Eg: “Marg”

There are some street names which are not part of the expected list in lesson 6 since these names are used locally in India. I simply added those names to the list of expected street names to solve this.

- Abbreviated street names. Eg: “Rd” for “Road”

I programatically replaced the abbreviations with the full street names.

- Street names with incorrect spelling. Eg. “Chauk” instead of “Chowk”

I programatically replaced the incorrect names with the correct names. Similar to the abbreviated street names.

- Street names that include the locality and city. Eg. “MG Lane, Andheri West, Mumbai”

To counter this I programatically extracted the characters before the first way. In this way I could extract the correct street names.

- Street names ending with integers. Eg: “Road No. 12”

Since these are actually valid street names, I manually included them in the list of expected street names.

- Completely invalid street names. Eg: “Andheri”

These are cases where a building name, locality, suburb or the city name is written as the street names. These corrections can’t be done programatically. To correct these names, we would need to check the lat long on the map and manually enter street names for all cases. Since this would have taken a huge amount of time, I ignored these street names (street name field was left blank) for the purpose of this project.

City

The city name should be one of Mumbai, Navi Mumbai or Thane since these three cities are part of the Mumbai Metropolitan Area. The common errors encountered were as follows.

- Incorrect spelling of city name. Eg: “Mumboi”

These incorrect spelling were replaced programatically using the same methods as that of abbreviated street names.

- Locality or neighbourhood name instead of city. Eg: “Colaba”

In these cases the locality/neighbourhood was replaced by the city name to which that locality/ neighbourhood belonged.

State

The state name should be Maharashtra for all cases. I only encountered one type of problem in a few cases.

- Incorrect spelling. Eg: “Maharshtra”

The incorrect spelling was replaced by the correct one.

Country

The country name should be India for all cases. There were only a few cases with the following problem.

- Abbreviated country name. Eg: “IN”

These cases were replaced by the correct country name.

2. Data Overview

This section contains some statistics about the dataset and the MongoDB queries used to collect them.

File Sizes

mumbai_india.osm	349.8 MB
mumbai_india.osm.json	528 MB

- Number of documents

```
> db.mum.find().count()
1912189
```

- Number of nodes

```
> db.mum.find({"type":"node"}).count()
1702595
```

- Number of ways

```
> db.mum.find({"type":"way"}).count()
209396
```

- Number of unique users

```
> db.mum.distinct("created.user").length
1151
```

- Total places of worship

```
> db.mum.aggregate([{“$match":{"amenity":{"$exists":1},
“amenity":"place_of_worship"}},
{"$group" : {"_id":"Total Places of Worship”,"count":{"$sum":1}}}])

{ "_id" : "Total Places of Worship", "count" : 394 }
```

- Most popular fast food joint

```
> db.mum.aggregate([{“$match":{"amenity":{"$exists":1},
“amenity":"fast_food"}},
{"$group" : {“_id":"$name","count":{"$sum":1}}},
{"$sort" : {"count" : -1}},{"$limit":1}])

{ "_id" : "McDonald's", "count" : 15 }
```

3. Other Ideas

In this section I will propose additional ways in which this dataset can be improved.

Using Google Maps API

The street address, postal code, city, state, country and other address related data can be pulled from Google Maps API using the lat long values as input. In cases where the OSM data is empty, the Google Maps API data can be used to complete it. And in cases where the OSM data is already present, the Google Maps API data can be used to verify the OSM data. In case there is clash between both data points, they can be evaluated programatically or manually (depending on the case) to decide on the correct data point. One issue that we could face doing this is faulty Google Maps data. So the other tests that we are applying to wrangle the data points should still be applied to Google Maps data. But this data could act as a good extra level of check to make OSM more accurate and clean.

Avoiding spelling mistakes

The spelling mistakes in city and state name can easily be avoided at the data entry stage on the OSM website itself. When I tried adding a node to OSM in Mumbai, I did not receive a suggestion for city, state or country name even though I has already selected the location and hence OSM knew the lat long. Since the boundaries are defined, the city, state and country name can be automatically pulled and given as a default suggestion to the user. This would lead to much less spelling mistakes. Since the user has the ability to overwrite the suggestion, wrong boundaries would not lead to any problems.