OpenStreetMap Data Case Study: Chennai, India

Map Area

For this project, I chose a bounded region from OpenStreetMap that is part of a suburban area located near Chennai, Tamilnadu, India (https://www.openstreetmap.org/relation/1766358). The osm file was obtained using the Overpass API. I chose this region as I grew up here; the insider knowledge helped me audit fields like zip codes and city names and knowing the local language helped me identify anglicized forms of the words describing roads, paths, etc.

The preprocessing was performed using the file main.py and the schema used is in schema.py.

Auditing and Updating the Map

The following problems were observed when auditing the file.

1. Abbreviations in street names

```
    <tag k="addr:housenumber" v="132"/>
    <tag k="addr:street" v="Arcot Rd"/>
    <tag k="amenity" v="place_of_worship"/>
    <tag k="name" v="Mata Amritanandamayi Math, Bhramastanam
Temple"/>
```

2. Spaces in zip codes

```
<tag k="addr:housenumber" v="Asvini Amarisa"/>
<tag k="addr:postcode" v="600 089"/>
<tag k="addr:street" v="kalasathamman koil st"/>
<tag k="building" v="yes"/>
<tag k="name" v="Asvini Amarisa"/>
```

3. Formatting inconsistencies in city name

```
d="322785" user="BCNorwich">
  <tag k="addr:city" v="chennai"/>
  <tag k="addr:housenumber" v="100A"/>
  <tag k="addr:postcode" v="600083"/>
  <tag k="addr:street" v="Babu rajendra
  <tag k="name" v="Rani Appartments"/>
</node>
<node id="3247846010" lat="13.0614534" l
rsion="1" timestamp="2014-12-22T15:04:22
d="2533271" user="DHANA STICKERS">
  <tag k="addr:city" v="CHENNAI"/>
  <tag k="addr:housenumber" v="NO301"/>
```

These were cleaned up appropriately by using the audit_update_element function.

1. Abbreviations in street names

```
"""
The validity of street name is checked and replaced from mapping if needed
"""
if key == "addr:street":
    value = element.attrib['v']
    if any(s in value for s in street_mapping.keys()):
        for label in street_mapping:
            if label in value.split():
                better_name = value.replace(label, street_mapping[label])
                print(value, "=>", better_name)
                element.attrib['v'] = better_name
                if not(all(s.isalpha() or s.isspace() or s.isdigit() for s in better_name) or (better_name in unique_street)):
                    unique_street.append(better_name)
```

A mapping from commonly found street abbreviations to their complete forms was used in this function. Furthermore, street name tags whose values appear to have multiple segments were maintained in a list and printed as not having been cleaned up.

2. Spaces in zip codes

```
"""
The validity of zip code is checked and spaces are eliminated if found
"""
if key == "addr:postcode":
    value = element.attrib['v']
    if (" " in value):
        print(value, "=>", value.replace(" ", ""))
        value = value.replace(" ", "")
        element.attrib['v'] = value
    if (value[0:3]!='600') or (not value.isdigit()) or (len(value) != 6):
        unique_zip_code.append(value)
```

The above piece of code was used to eliminate spaces from the zip code values. Moreover, all zip codes in Chennai begin with ‘600’ and have 6 digits, and this knowledge was used to create a list of zip codes that couldn’t be cleared up.

3. Formatting inconsistencies in city name

```
"""
The validity of city name is checked and formatting is updated if found
"""
if key == "addr:city":
    value = element.attrib['v']
    if (value != 'Chennai') and (value not in unique_city):
        if ('chennai' in value.lower()):
            print(value, "=>", "Chennai")
            element.attrib['v'] = "Chennai"
        else:
            unique_city.append(value)
```

The city values were checked for presence of the word “Chennai” and updated if needed. This also helped clear up the fields that erroneously included the suburb name in the city field.

```
<tag k="addr:city" v="Kolapakkam, Chennai"/>
<tag k="addr:postcode" v="600122"/>
<tag k="addr:street" v="II Avenue Maxworth Nagar Phase II"/>
```

Organizing osm content into csv files

The updated content was written into csv files using the shape_element function in the code provided in the data.py exercise with minor modifications; the schema used was obtained from the case study resources as well. Sqlite was for used for exploring the data in the csv files.

Data exploration using SQLite3

File sizes

```
Divya@Divya-laptop MINGW64 ~/Desktop/Nanodegree/Wrangling/Project
$ ls -l map.osm
-rw-r--r-- 1 Divya 197121 54140119 Nov 15 10:36 map.osm

Divya@Divya-laptop MINGW64 ~/Desktop/Nanodegree/Wrangling/Project
$ ls -l nodes.csv
-rw-r--r-- 1 Divya 197121 19881069 Nov 23 09:46 nodes.csv

Divya@Divya-laptop MINGW64 ~/Desktop/Nanodegree/Wrangling/Project
$ ls -l nodes_tags.csv
-rw-r--r-- 1 Divya 197121 88272 Nov 23 09:46 nodes_tags.csv

Divya@Divya-laptop MINGW64 ~/Desktop/Nanodegree/Wrangling/Project
$ ls -l ways.csv
-rw-r--r-- 1 Divya 197121 3301291 Nov 23 09:46 ways.csv

Divya@Divya-laptop MINGW64 ~/Desktop/Nanodegree/Wrangling/Project
$ ls -l ways_nodes.csv
-rw-r--r-- 1 Divya 197121 6951880 Nov 23 09:46 ways_nodes.csv

Divya@Divya-laptop MINGW64 ~/Desktop/Nanodegree/Wrangling/Project
$ ls -l ways_tags.csv
-rw-r--r-- 1 Divya 197121 1828047 Nov 23 09:46 ways_tags.csv

Divya@Divya-laptop MINGW64 ~/Desktop/Nanodegree/Wrangling/Project
$
```

Analyzing the number of elements in the nodes and ways tables

```
sqlite> select count(*) from nodes;
count(*)
--------------------
234912
```

```
sqlite> select count(*) from ways;
count(*)
--------------------
53357
```

The output from the table queries matched the number of nodes and ways obtained when profiling the map osm file using main.py.

```
Map structure
{'note': 1, 'meta': 1, 'bounds': 1, 'tag': 58381, 'node': 234912, 'nd': 290214, 'way': 53357, 'member': 7729, 'relation': 171, 'osm': 1}
```

Analyzing user statistics

The number of unique users who contributed to content in this region was obtained to be 243.

```
sqlite> select count(distinct(uid)) from
   ...> (select uid from nodes union all select uid from ways);
count(distinct(uid))
--------------------
243
```

The following are the top ten users in terms of number of contributions.

```
sqlite> select user, COUNT(*)
   ...> from (select user from nodes union all select user from ways)
   ...> group by user
   ...> order by count(*) desc
   ...> limit 10;
user                COUNT(*)
------------------  ----------
rajureddyvudem      47137
thrinath            27264
samuelmj            26383
saikumar            24915
maheshrkm           19372
ravikumar1          13724
Tinkle              12870
jasvinderkaur       12840
venkatkotha         11144
sdivya              9881
```

I was curious to know whether people only updated certain favorite locations or did they provide details about the paths between them as well. The following two queries showed that some people only uploaded information on nodes and some only described ways.

Number of users who contributed nodes information but not ways information

```
sqlite> select count(distinct(n.uid))
   ...> from nodes n left join ways w
   ...> on n.uid = w.uid
   ...> where w.uid is NULL;
count(distinct(n.uid
--------------------
86
```

Number of users who contributed ways information but not nodes information

```
sqlite> select count(distinct(w.uid))
   ...> from ways w left join nodes n
   ...> on w.uid = n.uid
   ...> where n.uid is NULL;
count(distinct(w.uid
--------------------
31
```

<u>Analyzing amenities in the region</u>

The top 10 amenities from the nodes_tag table and ways_tags table were queried.

```
sqlite> select value, count(value) from nodes_tags where key='amenity'
   ...> group by value
   ...> order by count(value) desc
   ...> limit 10;
value               count(value)
------------------  ------------
place_of_worship    41
school              27
atm                 23
hospital            18
fuel                17
police              14
restaurant          13
bank                11
fast_food           11
pharmacy            10
```

```
sqlite> select value,count(value) from ways_tags where key='amenity'
   ...> group by value
   ...> order by count(value) desc
   ...> limit 10;
value                count(value)
-------------------  ------------
school               16
place_of_worship     7
bus_station          5
hospital             5
restaurant           4
college              3
parking              3
marketplace          2
university           2
cinema               1
```

Places of worship, educational institutions (school, college, university), and healthcare facilities (hospital, pharmacy) seem to abound in this area. Next, a comprehensive list of amenities from both tables was generated.

```
sqlite> select distinct(value) from
   ...> (select distinct(value) from nodes_tags where key = 'amenity'
   ...> union all
   ...> select distinct(value) from ways_tags where key = 'amenity');
value
-------------------
clinic
bus_station
hospital
toilets
post_office
telephone
school
cinema
fuel
college
place_of_worship
fast_food
restaurant
police
public_building
atm
car_rental
bank
pharmacy
theatre
taxi
cafe
parking
post_box
Election Commission
kindergarten
waste_basket
fountain
townhall
driving_school
doctors
parking_space
bench
library
bar
community_centre
drinking_water
marketplace
social_center
university
```

Possible improvements to the dataset

An integrity check could be enforced on values entered for some fields; for example, a numeric check on door number, forbidding comma separated fields in street name, etc. This will help avoid introduction of errors such as the following where the street name has details about the housing complex and suburb as well.

```
Errors remaining in street name values
['Karumariamman Koil Street, Poonthottam Colony, Nandambakkam',
 'Jawaharlal Nehru Road, Balaji Nagar, Ekkatuthangal',
 'Govindasami Street, MGR Nagar, Nesapakkam']
```

This can also be implemented as a preprocessing function before accepting the values into the database.

```
sqlite> select substr(value, 1, instr(value, ',')-1)
   ...> from nodes_tags
   ...> where key = "street" and
   ...> value in ('Karumariamman Koil Street, Poonthottam Colony, Nandambakkam',
   ...>  'Jawaharlal Nehru Road, Balaji Nagar, Ekkatuthangal',
   ...>  'Govindasami Street, MGR Nagar, Nesapakkam');
substr(value, 1, instr(value, ',')-1)
------------------------------------------------------------
Karumariamman Koil Street
Jawaharlal Nehru Road
Govindasami Street
```

In terms of implementation, the integrity check would involve a warning message for a user who has entered incorrect text that explains the reason, for example, “Door numbers must be numeric.” An anticipated issue in this case could be that, in a particular region, door numbers might be alphanumeric or involve special characters. In Chennai, after a reassignment of door numbers by the corporation, the old numbers were phased out gradually and the interim door numbers were in the format of “old number/new number.” Such regional practices would have to be accounted for in the integrity checker, making it more complex to implement.

Further, there appear to be several synonyms used for the same amenities from the comprehensive list; example, cinema and theatre; clinic, hospital, and doctors; parking and parking_space etc. These values could be grouped under larger categories in processing so that a user requesting for information on hospitals would also be provided with information on clinics and doctors automatically. In terms of implementation, this would involve a preprocessing step in which a list of all the amenities is generated and someone has to manually group them. This would be expensive in terms of man hours and the cost might not be justified depending on how much of variability is seen in the tag values. Also, it is not clear if people realize the subtle difference between terms like hotels and restaurants, which are often used interchangeably in India. To avoid this, I would suggest having a predefined list of amenities that a user can select from when he enters a value. Any region-specific requests can be added on an ad-hoc basis.

References

1. Udacity’s course materials

2. Code from OpenStreetMap Data[SQL]

3. http://sqlite.org/lang_corefunc.html

4. StackOverflow solutions