Project-2: Wrangle OpenStreetMap Data

Project Submitted as part of AIRBUS Data Analyst Nanodegree.
by: Pankaj NATH on April 04, 2020 (Iss.1.1).

Table of Revisons

Issue	Date	Reason for Revision
1.0	04/04/2020	First version.
1.1	05/04/2020	Updated based on review conmments.

Table of Contents

Introduction: Map Area

This report is made towards project submission for Data Wrangling of OpenStreeMap data. I selected the Whitefield locality of East Bengaluru (previously named as Bangalore), a southern city of India which is also known as India's Silicon-Valley. The reason for selecting this map location is because I am residing in this part of the city form last 7 years. During this period of time, I had changed my office and appartment few times and has explored this neighbourhoods many times.
The objective of this project is to download map data from OpenStreetMap and apply different data wrangling techniques to audit data quality and improve data quality and parse the OSM map data into CSVs, importing CSVs to SQL tables. The improved data is then loaded into SQL database and queried to get other insights on it.
I have used python scripts to audit and clean OSM data and parse them into CSVs and creating database file. All these I have done using JupyterLab. This is an online platform for Jupyter Notebook. Later I imported the database file to SQL Online platform to execute my SQL quries. This is also another online platfor for SQLite. Usage of online platforms during this project work was really a very unique, smooth and hassele free experience for me.

Information on the map data used for the project:
Location: Whitefield, Bengaluru, India.
Download link: https://www.openstreetmap.org/export#map=12/12.9716/77.7472

This project includes below listed files:

1. **OpenStreetMap_DataWrangling.ipynb**: This Project report as a Jupyter Notebook.
2. **OpenStreetMap_DataWrangling.pdf**: This Project report in PDF format.
3. **map.osm**: Map data downloaded from OpenStreetMap.
4. **sample.py**: The script to make smaller extract from *map.osm*.
5. **smaple.osm**: Samller extract of *map.osm* made using *sample.py* on which data audit is performed.
6. **count_tags.py**: The script to parse *map.osm* to count unique tags in it.
7. **audit_city_postcode.py**: The script to audit city name and associated Indian Postal Codes.
8. **audit_street.py**: The script to audit street names.
9. **clean_city_street.py**: The script to update city and street names.
10. **clean_postcode.py**: The script to update postal codes.
11. **schema.py**: This is the schema for tables to be created.
12. **update_convert.py**: This scripts makes use of cleaning and schema scripts to clean original OSM data and creates CSV tables.
13. **create_db.py**: The script to create databse file from CSV files.
14. **map.db**: This is the database file made using CSV files and *create_db.py*.

Problems Encountered in the Map

Data Audit

By parsing through **map.osm** data using **Count_tags.py** script, below result is obtained for count of unique tags in my map data:

{'bounds': 1, 'member': 22462, 'meta': 1, 'nd': 627068, 'node': 514183, 'note': 1, 'osm': 1, 'relation': 464, 'tag': 178804, 'way': 111156}

Using the **sample.py**, a smaller extract (*sample.osm*) is created from original data (*map.osm*). This is done for quick auditing of map data by saving time of execution run on larger map data. The sample extract is created by extracting every 50th element from original map data. This sample.osm data is further used for auditing.

In India the address writing format is not as standardized as in other developed countries. The addresses in India are usually very long and does not have same format throughout the country. For the map zone of Whitefield, Bengaluru, it is true that a street name, city name and a postal code as per locality is a must. Using **audit_city_postcode.py** and **audit_street.py** two dictionaries of different city names & their postal code and street names is created.

The postal codes in India are of 6-digits and for the map zone of Whitefield area it shall be in 560xxx format. From the audit of postal code, I identified that the old city name ***Bangalore*** is still used and not the new name ***Bengaluru***. I propose to update the city name and harmonize the postal code.

The harmonization of postal code and update of city and street names is achieved programatically using **clean_postcode.py** and **clean_city_street.py** scripts respectively.

City Name Corrections

The city name is presnt in OSM data with tag k= 'addr:city'.

1. 'Bangalore' -> 'Bengaluru'
2. 'bangalore' -> 'Bengaluru'
3. 'bengaluru' -> 'Bengaluru'

Postal Code Harmonization

The city name is presnt in OSM data with tag k= 'addr:postcode'.

1. If postal code is empty then no action.
2. If postal code does not begin with '560' then make them empty.
3. If postal code is not of 6 digits then make them empty.
4. If postal code is of 6-digits begining with '560' then no action.

Street Name updates

The city name is presnt in OSM data with tag k= 'addr:street'.

1. Updating the first letter in the name spellings to uppercase.
2. Delete all special characters from street names.
3. Updating abbreviation from 'Rd'/'Rd.' to 'Road', 'Ft'/'Ft.' to 'Feet' and other similar updates.

Once all cleaning is performed on **map.osm** data, XML data is converted to following five CSV tables using **update_convert.py** and **schema.py**:

> 1. **nodes.csv**
> 2. **nodes_tags.csv**
> 3. **ways.csv**
> 4. **ways_nodes.csv**
> 5. **ways_tags.csv**

Above five CSVs were then used to build **map.db** database file. This database file is then used in SQLite.

Data Overview

This section of the reports provides some statistical overview on the files size and other informations extracted from the database with SQL querries.

File sizes

1. **map.osm**: 113 MB
2. **sample.osm**: 2.29 MB
3. **nodes.csv**: 40.9 MB
4. **nodes_tags.csv**: 1.4 MB
5. **ways.csv**: 6.43 MB
6. **ways_nodes.csv**: 14.3 MB
7. **ways_tags.csv**: 4.55 MB
8. **map.db**: 76.9 MB

Number of Unique Users

With below query, Number of unique users contributing to map data is **1157**.

```
SELECT COUNT(distinct(uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways)
```

Number of Nodes

With below query, Number of nodes in map data is **514183**.

```
SELECT COUNT(*) FROM nodes
```

Number of Ways

With below query, Number of ways in map data is **111156**.

```
SELECT COUNT(*) FROM ways
```

Top Postal Code

With below query, top 3 most used postal codes from map data is queried as:

Post Code	Count
560037	209
560066	202
560103	140

```
SELECT n.value, COUNT(*) as count
FROM
(SELECT * FROM ways_tags UNION ALL
```

```
SELECT * FROM nodes_tags) n
WHERE n.KEY='postcode'
GROUP BY n.value
ORDER BY count DESC
LIMIT 3
```

Top Amenities

With below query, top 5 amenities from map data is queried as:

Amenity	Count
restaurant	648
atm	223
bank	215
place_of_worship	213
fast_food	211

```
SELECT value, COUNT(*) as count
FROM nodes_tags
WHERE KEY='amenity'
GROUP BY value
ORDER BY count DESC
LIMIT 5
```

Additional Ideas

City and Street names in address

I observed that many of the street name fields contain the complete address including city name and postal code. I propose additionally we can identify such entries in map data and remove city name and postal code from street name field. If not present then, city name and postal code can be added as new tags for those nodes.

Benefit(s):

This will bring some uniformity and standardize the address throughout the data. Not to forget, manual data reading and interpretation will improve as well.

Anticipated Problem(s):

The proposed solution should take care of all scenarios possible before removing user provided city names and postal codes. This solution only reorganizes and makes the user input data more readable but it doesn't gaurantees that the city name and postal code would be correct!

Postal Code

I assumed first 100 postal codes to be present in my map data during this project. As an additional idea, I propose to make use of Indian Postal departments complete list of postal code extract from their website which also contains city or locality name. Then make use of this extract to update postal code in our map data corresponding to its steet, locality or city name.

Above idea is supported by the fact that below query gives three postal codes **570008, 625009 and 530103**. None of these begin with '560' as expected.

```
SELECT DISTINCT(value) FROM nodes_tags
WHERE key='postcode' AND value NOT LIKE '560%'
```

With a google search using keyword *pincode 570008*, I found this code belongs to Mysore (Mysuru, new name) which is a different city altogether. Similarly *625009* belongs to Madurai city which is in a different state altogether and *530103* is an invalid code which doesn't exist.

Benefit(s):

Implementation of this solution will remove all wrong post codes and also it can fulfill the post codes which are missing based on the city or locality names (if present).

Anticipated Problem(s):

If the user provided city or locality name is incorrect then this could lead to wrong post code assignment. Hence the solution should also be smart enough to check nearby post codes before taking any decision. Building such a smart solution may be require additional efforts but return of value may be insignificant.

Tags

I observed that some nodes have very few tags and others have more. This discrepancy can be further checked and if some key places are missing these tags then they can be fulfilled as well.

Benefit(s):

Implemention of such a solution will bring completeness in map data.

Anticipated Problem(s):

It could be possible that not all tags are needed for any particular location, this may then lead to too many empty tags associated to all location. It may then impact the size of map data by increasing its size without much value addition. Hence probably we should identify the node types to categorize them and then allocated minimum tags which much be present for each particular node type.

Conclusions

The OpenStreetMap data for Whitefield, Bengaluru is not uniform throughout. There are above 1000 users who contributed to this map data.
As already discussed the problems with address format in India, manual entry of data further pollutes the data. As a part of this project, it is demonstrated that such cleaning is possible and

by working on additional ideas proposed above, higher quality of data can be achieved.
As another measure, OpenStreetMap can put in place some control measures so that users do not intensionally pollutes the common tags like city name and postcodes.