OpenStreetMap Project

Area: I have chosen Bengaluru, India to perform my analysis because it is my hometown. I used Map Zen to extract the OSM file.

Data Auditing

I use the parser.py file to count the number of unique tags in the Bengaluru_india.osm file. I did this by parsing the file using ElementTree. The count of the number of unique tags is;

- 'bounds': 1,
- 'member': 8529,
- 'nd': 3612323,
- 'node': 2915140,
- 'osm': 1,
- 'relation': 1240,
- 'tag': 833064,
- 'way': 664256

Next, I used tags.py to create 3 regular expressions to check for patterns in the tags. The "k" value of each tag contains different patterns. I have counted all of the 4 categories;

- 'lower': 789410, for tags that contain only lowercase letters and are valid.
- 'lower_colon': 42513, for tags with a colon in their names
- 'other': 1139, for tags that cannot be otherwise categorized
- 'problemchars': 2, for tags with problematic characters.

Problems encountered in the map

Using audit.py, I updated some of the common mistakes found in the dataset. The main mistakes were inconsistencies in the names. Some of them are;

- Rd => Road
- temple => Temple
- Bangalore => Bengaluru
- Ft. => feet
- KA => Karnataka

Data Overview

- bengaluru_india.osm – 656.5 MB
- nodes.csv – 244.2 MB
- nodes_tags.csv – 3.9 MB
- ways.csv – 40.5 MB
- ways_nodes.csv – 86.7 MB
- ways_tags.csv – 24.4 MB
- bengaluru.db – 455.7 MB

For the following queries, I first executed database.py to create the database and queries.py to run the following queries;

Number of nodes:

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

2915140

Number of ways:

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

664256

Number of unique users:

```
SELECT COUNT(DISTINCT(e.iud)) FROM
(SELECT uid FROM nodes UNION ALL
SELECT uid FROM ways)
```

2057

Top contributing users:

```
SELECT e.user, COUNT (*) as num FROM
(SELECT user FROM nodes UNION ALL
SELECT user FROM ways)
GROUP BY e.user
ORDER BY num DESC LIMIT 10
```

Jasvinderkaur – 124827
Akhilsai – 118664
Premkumar – 115877
Saikumar – 114883
Shekarn – 98110
PlaneMad – 94898
Vamshikrishna – 94251
Himalay – 88139
Himabindhu – 86840
Sdivya - 84980

Number of users contributing only once:

```
SELECT COUNT (*) FROM
(SELECT e.user, COUNT (*) as num FROM
(SELECT user FROM nodes UNION ALL SELECT user FROM ways)
GROUP BY e.user HAVING num = 1)
```

516

Additional Data Exploration

Common Ammenities:

```
SELECT value, COUNT (*) as num FROM nodes_tags
WHERE key = “amenity”
GROUP BY value
ORDER BY num
DESC LIMIT 10
```

Restaurant – 1721
Atm – 841

Bank – 785
place_of_worship – 722
fast_food – 570
pharmacy – 569
hospital – 460
school – 381
café – 348
fuel - 287

Biggest Religion:

```
SELECT nodes_tags.value, COUNT (*) as num FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value = “place_of_worship”
ON nodes_tags.id = i.id
WHERE nodes_tags.key = “religion”
GROUP BY nodes_tags.value DESC
LIMIT 1
```

Hindu – 455

Popular Cuisines:

```
SELECT nodes_tags.value, COUNT (*) as num FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value = “restaurant”
ON nodes_tags.id = i.id
WHERE nodes_tags.key = “cuisine”
GROUP BY nodes_tags.value
ORDER BY num
DESC LIMIT 10
```

Regional – 328
Indian – 252
Chinese – 76
Vegetarian – 58
Pizza – 36
International – 30
Italian – 27
Burger – 12
ice_cream – 12
Andhra - 11

Conclusion

The OpenStreetMap data of Bengaluru is of pretty big and has fairly reasonable quality. Of course, since the data is human input, there are a lot of typos and errors. I cleaned the possible typos but, there is much more work need to be done. The dataset contains a lot of information on Bengaluru and is most likely not up to date. Hence, I think there are a few areas of improvement despite being such a large dataset.

Additional Suggestion and Ideas

Control typo errors

- We can parse the words input by users by building a parser.
- We can have a pre-set syntax for users to input their data in a specified way.
- We can periodically clean errors by automating a programming check.

More information

Since, major attractions and popular destinations are important data to a city, users must be encouraged to provide more information of the same to keep the data updated. This can help increase the viewership of the maps.

Anticipated problems

The possible problems that we could face with my suggestions is that;

1. We will need a complex parser to process such a large amount of data and the similar words can have duplication.
2. Updating data on fast – growing metropolitan cities is hard.

Files

bengaluru_india.osm: city data as an OSM file
parser.py: counts tags
tags.py: counts patterns
audit.py: audit street, city and update their names
schema.py: defines the structure of the data
data.py: build CSV files from OSM and also parse, clean and shape data
database.py: create database of the CSV files
query.py: different queries about the database using SQL

References

- Udacity OpenStreetMap Case Study
- https://github.com/pratyush19/Udacity-Data-Analyst-Nanodegree/tree/master/P3-OpenStreetMap-Wrangling-with-SQL