Wrangle Open Street Map Data

I. Map Area

Ahmedabad, Gujarat, India : Ahmedabad, Gujarat is my native place and I visit the city often.

- Ahmedabad on OpenStreetMap
- Ahmedabad on Mapzen

II. Data Auditing

Unique Tags
Using iterative parsing, we process the map file and find out how many individual tags are there. We will be using **mapparser.py** for this purpose.

- 'bounds': 1
- 'member': 2267
- 'nd': 639102
- 'node': 550116
- 'osm': 1
- 'relation': 504
- 'tag': 99384
- 'way': 82273

We see from above that maximum number of tags in our OSM is 'nd' tag followed by the 'node' tag.

Tag categories
Based on the 'k' value for each '<tag>' tag, we divide the '<tag>' tags, into 4 categories:

- **"lower"** , for tags that contain only lowercase letters and are valid,
- **"lower_colon"** , for otherwise valid tags with a colon in their names,
- **"problemchars"** , for tags with problematic characters, and
- **"other"** , for other tags that do not fall into the above three categories.

In **tags.py** , 3 regular expressions have been compiled to check and count for above categories in the tags.

- 'lower': 97326
- 'lower_colon': 2017
- 'other': 34
- 'problemchars': 7

III. Problems Encountered in the Map

In total, 3 problems were encountered in the map data.
Old names were corrected with the updated names and have been shown below using ->. Left side of the arrow indicated the old name and right side of the arrow indicates the updated name.
Using **audit.py**, we corrected the problems.

Street address inconsistencies
There were 2 main inconsistency problems in the dataset which are listed as below.

- **Abbreviations in the street name:**
  Rd --> Road
- **lowercase letters in place of UPPERCASE letters**
  sbk -> SBK
  gandhi -> Gandhi
  ahmedabad -> Ahmedabad

Wrong spellings

socity -> Society
Ahmadabad -> Ahmedabad

Names in local language

rasta -> Road
char -> Four

IV. Data Overview

We will build some useful CSV files from OSM and also parse, clean and shape our data using **data.py.**

We then create a database from these CSV files using **db.py.** We shall run our queries on this created db file using SQLite.

File sizes

1. ahmedabad_india.osm: 109 MB
2. nodes.csv: 43.9 MB
3. nodes_tags.csv: 0.248 MB
4. ways_csv: 4.76 MB
5. ways_nodes.csv: 15.3 MB
6. ways_tags.csv: 2.96 MB
7. ahmedabad.db: 78.7 MB

Number of unique users

sqlite> SELECT COUNT(DISTINCT(e.uid))
FROM
(SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
Output:
369

Number of nodes

sqlite> SELECT COUNT(*) FROM nodes
Output:
550116

Number of ways

sqlite> SELECT COUNT(*) FROM ways
Output:
82273

Common amenities

sqlite> SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 3;
Output:
"place_of_worship" "69"
"restaurant" "49"
"hospital" "33"

V. Additional Statistics

Top contributing users

sqlite> SELECT e.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
GROUP BY e.user
ORDER BY num DESC
LIMIT 10;
Output:
"uday01" "177284"
"sramesh" "136707"
"chaitanya110" "122312"

Number of trees

sqlite> SELECT COUNT(*) as num
FROM nodes_tags
where value like '%tree%';
Output:
574

Most prominent religion

sqlite> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE
value='place_of_worship') i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='religion'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 1;
Output:
"hindu" "34"

VI. The Conclusion

The OpenStreetMap data of Ahmedabad has many errors caused by the human inputs which are significant. We have cleaned a significant amount of the data which is required for this project. But, there is still scope of improvement over the dataset. The dataset contains very less amount of additional information such as amenities, tourist attractions, popular places and other content as compared to that of Google Maps. So, I think there are several opportunities for cleaning, validation and updation of the data in the future.

VII. Additional Suggestion and Ideas

Control typographical errors

- A parser can be built which parses every word which is input by the users.
- Some stringent rules can be made and enforced on input data which users must follow everytime to input their data. This will also restrict users to give input in their native language.
- Scripts can be created and put on scheduler to clean the data regularly.

Possible improvements

The tourists, sometimes even the native people, search maps for basic amenities, popular places and attractions to visit in or near the city, best restaurants based on user ratings, hotels to stay based on the tourist's budget. So, the users must be motivated to also submit such information in the map thereby increasing the chances for tourists to use OpenStreetMaps.

Challenges in implementing additional ideas

The only challenge in implementing the solution for controlling typographical errors and possible improvements is **complexity**.
It will be difficult and complex to build a parser which parses **every word** and the rules to be enforced on input data must be updated regularly.