Open Street Map Data Case Study

Map Area

The case study is performed on a square area bounded by longitudes (73.160 - 79.365) and latitudes (24.277 - 28.150).

Location: Jaipur,India

Problems Encountered in the Map

- Two different ‘k’ attributes for postcodes:

'addr:postcode': {'284001', '311001', '333012'},
'postal_code': {'202124', '305403', '323001', '325220'},

I have tried to solve this problem in create_csv.py by using the code in shape_element() function:

```
if child.attrib['k'] == 'postal_code':
        node_tag['key']='Postcode'
        node_tag['type']='Addr'
```

- Over-abbreviated ‘k’ attributes:

'name:hi' , 'name:kn' , 'name:ta', ‘name:ru’ etc.

I have tried to solve this problem by creating a mapping dictionary in create_csv.py and creating a update_name() function and using it in shape_element () function.

```
def update_name(name, mapping):
   a=name[(name.index(":")+1):]
   if a in mapping.keys():
      name = name.replace(a, mapping[a])
   return name
```

- Inconsistent values :

In the dataset where ‘k’ attribute is ‘name’ the values of attribute ‘v’ (supposed to have place names) has following inconsistencies:

1. Includes code digits :
   '026', '021','028','SS04ST - RTS;', '505 - OT;' etc.
2. Include Phrases :
   'Nuh', Unit 3' etc.
3. Person names:
   ‘ Dr.Rekha Gupta’, ’my home babita' etc.
4. Unicode charaters:
   劍南山流動廁所

Data Overview and Additional Ideas

This section contains basic statistics about the dataset, the sql queries are used to gather them, and some additional ideas about the data in context.

File sizes

Jaipurmap.osm	421 MB
database.db	239 MB
nodes.csv	171 MB
nodes_tags.csv	4.42 MB
ways.csv	9.95 MB
ways_tags.csv	8.08 MB
ways_nodes.csv	60.9 MB

Data Statistics using Sql Queries

Number of nodes:

Sqlite3> SELECT COUNT(*) FROM nodes

Output:
2,172,782

Number of ways:

Sqlite3> SELECT COUNT(*) FROM ways

Output:
168,696

Number of unique users:

Sqlite3> SELECT COUNT(DISTINCT(e.uid))

FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;

Output:
911

Top contributing users:

Sqlite3> 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:

Heinz_V	971,388
Oberaffe	249,402
Seandebasti	110,895
indigomc	81,297
bdiscoe	70,038
harishkonda	62,348
parambyte	41,563
kalekhya	39,673
chandusekharreddy	37,399
Ranjana Devanand	32,236

Number of users contributing only once:

```
Sqlite3> SELECT COUNT(*)
FROM
   (SELECT e.user, COUNT(*) as num
    FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
    GROUP BY e.user
    HAVING num=1) u;
```

Output: 210

Addition Statistics:

Common ammenities:

```
Sqlite3> SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='Amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

Output:

Place Of Worship	316
Fuel	262
Restaurant	249
Atm	140
Bank	133
Hospital	119
School	99
College	57
Cafe	51
Bus Station	45

Biggest religion:

```
Sqlite3> 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 : 105

Popular cuisines

```
Sqlite3> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
   JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='Restaurant') i
   ON nodes_tags.id=i.id
WHERE nodes_tags.key='Cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC
Limit 4;
```

Output:

Indian	23
Regional	4
Pizza	3
Bakes	2

Additional Suggestions

- Improving Contribution:
  - ❖ The total number of contributions to the dataset I used is 2,341,478.
  - ❖ The percentage of contribution by top contributor (‘Heinz_V’) is 41.48%.
  - ❖ The combined contribution of top 10 contributors is 72.44%.

  Thus we can see the contribution of users is skewed. We can have more contributions to Openstreet maps through Cab Services.
  People wanting to get more efficient service can submit route to destination from the point of pickup. This will add a significant data to open street maps which will be more accurate as the users would want to be dropped off at the most precise location.
  The drawbacks of this approach is that people travel more to some places while few places may be left undiscovered where the cab service is not available .

- For ‘k’ attribute equals to ‘addr:housenumber’ the values of ‘v’ attribute is not uniform. As it is human edited data , there are many places where we can see that house-number is not actually a house number.

  Sqlite3>SELECT value FROM ways_tags WHERE key=’Housenumber’ LIMIT 30;

  The Output of the above query will give us values which we can categories as:
  - D-27, 74B , 10 ,Sp-43, C-22/1 etc. (acceptable house-numbers)
  - Dhinwa ki Dhani, Rajeev Nagar, Opp Fortis Hospital etc. (strings which are not house numbers).

  This problem can be corrected by using regular expression which will pull out all the strings which does not have any integer value in it as a house-number must consist of a number and may have a block-code .Also the input data-type can be fixed to prevent insertion of harmful data. For example if postal code column is made to accept only integers type data then no user can input strings.

Conclusion

We have successfully used data munging techniques, such as assessing the quality of the data for validiy, accuracy, completeness, consistency and uniformity to clean the OpenStreetMap data for a part of world that I most care about. I have successfully imported data to SQL database according to a chosen schema and using SQL queries I have analysed the data. The map data still have some inconsistencies which require some gold standard data to clean it. But for the project a fair amount problems with the data has been removed.