.

Data Wrangling with MongoDB: OpenStreetMap in Kolkata, India

Author: Samit Chaudhuri

Overview

This project is about wrangling the Mapzen metro-exracts of OpenStreetMap data for the Kolkata metro-area in India [1]. Data from a different geographical region contains a lot of discrepancies with how such data is collected and described in the US. Therefore it is expected to be an interesting and challenging data wrangling exercise.

Code used in this report are included in the attached file "openstreet_kolkat.py".

Section 1. Problems Encountered in the Map Data

After loading the map data for Kolkata, India I first ran some audit code (e.g. functions audit_user_names(), audit_street_types() and audit_city_names() in openstreet.py) to list the user names, city names, and street names to assess the general goodness of data. The results with of the audit is presented along with the cleaning plan later in this section.

A cleaned up version of the map data should improve the five formal measurues of data quality including validity, accuracy, completeness, consistency, uniformity.

To improve the first of the five formal measures of data quality, *validity*, I would have prefered to start with a standard schema for Indial postal addresses. Unfortunately as discussed in [2], address format in India isn't very standardized. While US addresses contain a standard street/city/state format, Indian addresses often contain locality/town/city format, which makes it hard to fit into the fields offered in the osm format.

Auditing and deciding on the cleaning plan took serveral iterations between them. These iterations were performed outside before entering the data elements into MongoDB. It was faster to uncompress the .bz2 file once onto the disk, and then to repeatedly audit the data as they came out of the SAX parser.

Auditing and cleaning up contibuting user information

- A few users have contributed most of the data.
- Some user names TA few users have contributed most of the data.

Auditing and cleaning up local street types and names

The audit_street_types() function helped locate the following anomalies in the street names.

- Street types are not described in a consistent manner; for example Rd., Rd, road, Road.
- Some street types contain typing errors; for example, raod instead of road.
- Some street names contain house number information; for example "24/j, shyamsundar pally". These items can be made more accurate by removing the house numbers and storing them in the addr:housenumber field.

- Some of the street types are local to the area, such as "Sarani", "Pally", and "Potti".
- One street names contains an entire address "Plot No. X1-1, Block EP, Sector V, Saltlake, Kolkata - 700091".
- Some street names contain multiple street names; for example "41, Jawaharlal Nehru Road, Middleton Row, Middleton St".
- Some of the street names contain locality and town information; for example "Major Arterial Rd, Action Area IID, New Town".

The fix_street_name() function addresses the first 4 categories of issues. The remaining categories are related to accuracy, and need some gold standard data which are not available in this project.

Auditing and cleaning up city names

The audit_city_names() function helped identify following problems with the city names:

- Some city names are written differently in two different items; for example 'Salt Lake (Bidhan Nagar)', and 'Saltlake (Bidhannagar)'. For better data consistency, they should both be written in the same way, e.g. Salt Lake (Bidhannagar)
- Some city names are written with different capitalizations; for example 'kolkata', and 'Kolkata'. For better data consistency, they should both be written in the same way; e.g. Kolkata.

The fix_city_name() function cleans up the data and improves its accuracy by making the corrections mentioned above.

Auditing and cleaning up postal codes

Addresses in India use 6-digit postal codes. The data items use 41 unique 6-digit postal codes. The audit_postcodes() function helped identify following problems with the postal codes:

- There are two 4-digit postal codes 7000, and 7400. This can be an issue of data validity (is it a valid postal code in Kolkata, India ?), or data accuracy (does this postal code exist at all ?) issue. Google search reveals that the 7400 postal code is a correct postal code in neighboring country Bangladesh, and indeed the particular address is for a location in Bangladesh.

Some of the problems encountered during data audit are cleaned programmatically.

Section 2. Overview of the Data

A statistical overview about the data set is given below.

- Size of the file
    - kolkata_india.osm ... . 106 MB
    - kolkata_india.osm.json ... 155 MB
- Number of unique users

```
uniq_user_count = len(db.maps.distinct("created.user"))
print "There are {} of unique contrbuting users in Kolkata, India.'

There are 227 of unique contrbuting users in Kolkata, India.
```

- Number of nodes and ways

```
node_count = db.maps.find({"type" : "node"}).count()
ways_count = db.maps.find({"type" : "way"}).count()
print "There are {} nodes and {} ways in Kolkata, India.".format(no

There are 506727 nodes and 59642 ways in Kolkata, India.
```

- Number of chosen type of nodes, like cafes, shops etc.

```
cafe_count = db.maps.find({"amenity" : "cafe"}).count()
restaurant_count = db.maps.find({"amenity" : "restaurant"}).count(
shop_count = db.maps.find({"amenity" : "shop"}).count()
hospital_count = db.maps.find({"amenity" : "hospital"}).count()
school_count = db.maps.find({"amenity" : "school"}).count()
college_count = db.maps.find({"amenity" : "college"}).count()
univ_count = db.maps.find({"amenity" : "university"}).count()

print """Amenities:
    cafes: {}
    restaurants: {}
    shops: {}
    hospitals: {}
    schools: {}
    colleges: {}
    universities: {}
""".format(cafe_count, restaurant_count, shop_count, hospital_count

Amenities:
        cafes: 9
        restaurants: 38
        shops: 0
        hospitals: 75
        schools: 135
        colleges: 65
        universities: 25
```

Section 3. Other ideas about the datasets

Quality analysis of the openstreet data can be a rich topic. Some of the interesting concerns about such data are

- Is the data spatially accurate ?
- Is it always legal to use such data ?
- Can this data violate national security ?
- Given the level of accuracy, are the risks acceptable for a certain application.

One way to improve the accuracy of the data would be to establish a list of "ratings" for each data item, where other reviewers can rate the accuracy of each entry. This would make it possible to establish a quntifiable measure of fidelity of a data item based on number of reviews and their actual ratings. Applications will be able to filter out the data items that fall below some fidelity threshold. National security interests can be safeguarded by adding a "contributor-id" to each data item. This will allow regulatory agencies to perform security audit of sensitive areas and track down contributors that enter data that violate a nation's security interests.

If the quality is acceptable, the OpenStreetMap data can a gold mine of opportunities for many applications

- Real time routing projects e.g. open source routing machine [4]
- Provide spatial awareness to humanitarial projects that distribute goods and services to remote communities.
- Analysis on spatial distribution of land and resources.

Additional exploration using MongoDB queries

- Top 10 businesses

```
pipeline = [
    temp
        {"$match" : {"type" : "node",
                 "shop" : {"$exists" : 1}}},
    {"$group" : {"_id" : "$shop",
                 "count" : {"$sum" : 1}}},
    {"$sort" : {"count" : -1}},
    {"$limit" : 10}
]
result = db.maps.aggregate(pipeline)
print "Top 10 businesses:"
pprint.pprint(result)

Top 10 businesses:
{u'ok': 1.0,
 u'result': [{u'_id': u'supermarket', u'count': 15},
             {u'_id': u'convenience', u'count': 6},
             {u'_id': u'hairdresser', u'count': 5},
             {u'_id': u'car', u'count': 3},
             {u'_id': u'bakery', u'count': 3},
             {u'_id': u'electronics', u'count': 3},
             {u'_id': u'books', u'count': 3},
             {u'_id': u'alcohol', u'count': 3},
             {u'_id': u'clothes', u'count': 3},
             {u'_id': u'shoes', u'count': 3}]}
```

- Number of different types of highways

```
pipeline = [
    {"$match" : {"type" : "way",
                 "highway" : {"$exists" : 1}}},
    {"$group" : {"_id" : "$highway",
                 "count" : {"$sum" : 1}}},
    {"$sort" : {"count" : -1}},
    {"$limit" : 10}
]
result = db.maps.aggregate(pipeline)
print "Number and types of highways"
pprint.pprint(result)

Number and types of highways
{u'ok': 1.0,
 u'result': [{u'_id': u'service', u'count': 17201},
             {u'_id': u'residential', u'count': 14781},
             {u'_id': u'tertiary', u'count': 2257},
             {u'_id': u'unclassified', u'count': 680},
             {u'_id': u'secondary', u'count': 626},
             {u'_id': u'primary', u'count': 519},
             {u'_id': u'footway', u'count': 261},
             {u'_id': u'trunk', u'count': 248},
             {u'_id': u'track', u'count': 238},
             {u'_id': u'path', u'count': 114}]}
```

References

[1] Metro Extracts, City-sized portions of OpenStreetMap, served weekly

[2] India Mailing Address Formats and other International Address Information.

[3] OpenStreetMap Sample Project, Data Wrangling with MongoDB, Matthew Banbury

[4] Open Source Routing Machine, Wikipedia