OpenStreetMap Project
Data Wrangling with MongoDB

-Kiran Hegde

Map Area: Bengaluru, India.

https://mapzen.com/data/metro-extracts/metro/bengaluru_india/

1. Problems Encountered in the Map

After downloading the Bengaluru dataset and reducing it to a sample dataset containing one tenth the data to easily work with, it was run against various exploratory functions in the provisional osm_exploring.py file. The most prominent issues encountered were as follows:

1. Inconsistent street address fields:

The assumption that there isn't any particular convention of naming street addresses in India and hence Bengaluru, was confirmed after looking at the street name field entries in our dataset. Because of this reason, all the entries fall under the valid category. However, inconsistencies in the words used in naming street addresses like Rd., road, Road and bellandur, Bellandur and many more were found. Some of these were corrected in a way that best represents the street address to provide slightly better harmony in our data.

2. Multiple format of name field entries:

While exploring the name field, entries like "reliance foot prints", "Reliance Footprints" and other such multiple formats of names for the same brand of entities accross the map area was observed. Many of such names were synchronized with one appropriate name which best relates to the entity in context.

3. Wrong level field entries:

Some documents containing level field contained values like "0,1,2,3,4,5,6,7,8,9". These kind of values were used to describe levels of nodes or ways like building or apartments having more than one floor (10 floors in the case of the given example). According to the OpenStreetMap documentation, the level field entries in such a case are to be formatted like "0;9". Hence, level fields containing such values were cleaned to best follow the prescribed format.

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

-File sizes

bengaluru_india.osm . . . . . 639.9 MB
bengaluru_india.osm.json . . . . . 993.3 MB

-Number of documents

```
>db.bengaluru.find().count()

3498624
```

-Number of nodes and ways

```
> db.bengaluru.aggregate([{"$group": {"_id": "$type", "count": {"$sum": 1}}}])

{ "_id" : "way", "count" : 652928 }
{ "_id" : "node", "count" : 2845696 }
```

-Number of unique users

```
> db.bengaluru.distinct("created.user").length

1621
```

-Cuisines sorted by count

```
> db.bengaluru.aggregate([{"$match": {"cuisine": {"$exists": 1}}}, {"$group": {"_id":
"$cuisine", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}])

{ "_id" : "regional", "count" : 353 }
{ "_id" : "indian", "count" : 301 }
{ "_id" : "vegetarian", "count" : 92 }
{ "_id" : "chinese", "count" : 83 }
{ "_id" : "pizza", "count" : 74 } . . . .
```

-Amenities sorted by count

```
> db.bengaluru.aggregate([{"$match": {"amenity": {"$exists": 1}}}, {"$group": {"_id":
"$amenity", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}])

{ "_id" : "restaurant", "count" : 1607 }
{ "_id" : "place_of_worship", "count" : 957 }
{ "_id" : "atm", "count" : 708 }
{ "_id" : "bank", "count" : 683 }
{ "_id" : "school", "count" : 669 }
{ "_id" : "hospital", "count" : 554 } . . . .
```

3. Additional Ideas

1. Suggestions for reducing the input of wrongly formatted entries:

Looking into the opening_hours field, I noticed a lot of differences in the various formats used. However, the OpenStreetMap's prescribed format for opening_hours field is very simple and easy to use. This probably is not noticed by the users editing map probably because looking into the prescribed formats for every little field is too much to ask for? In the context of this assumption about the entry of wrongly formatted values, the interface of the OpenStreetMap could display the right format for the newly selected field in a way that is simple and easy to understand while the entry is being made. I also noticed the simplicity of OSM prescribed formats for other fields like level, etc. Hence, this approach can be used to fields that have prescribed formats. This i believe can reduce the recording of dirty data by reducing the wrong formats used while entering the data.

2. Benefits and anticipated problems in implementing the suggested improvements:

The most prominent of the benefits that can be expected are of cleaner data entering the database, which thus would reduce the amount of cleaning that has to be done prior to the analysing the datasets. The improvements discussed might also make it easier for the users editing map data which inturn might increase their effort towards editing more often. Although the suggested improvements don't demand anything even remotely close to a complete overhaul, it asks of creatively integrating the improvements with the existing framework. Hence, some of the problems that can be anticipated are of coming up with a thoughtful design for displaying the relevant information about the prescribed formats and interestingly integrating it with the OpenStreetMap interface such that the design in context is very simple and easy to understand without demanding too much time for enquiry.

3. Conclusion:

After reviewing the amout of data being recorded due to users' efforts, the potential of this data in various applications is huge. However, it is limited by the quality of data. Although processes for cleaning can be used to clean data, the most effective approach would be to minimize the entry of bad data at the source. Since the variety of sources are known, effort towards understanding the cause of bad data would thus be very beneficial towards efficiently using the resources being gathered in the form of data..