OpenStreetMap Project

Data Wrangling with MongoDB

Pradyut Vatsa

Map Area: City Bengaluru, Bangalore Urban, Karnataka, 560001, India

Map Link:

http://www.openstreetmap.org/node/3401391999

Why this region?

I'm from Bengaluru and typically map plots and anchors aren't very accurate here. I wanted to validate and be a part of the open source contribution project and understand how it is making a difference in the area where I stay. For this very reason, I've decide to use this as my selected area.

Citations

Only lecture videos and python and mongodb documentation used as reference. No other websites, blogs, github repositories used for making this project.

Problems encountered in the map

After downloading the entire map of size 600 MB, I took every 5th top level element of the map to investigate potential problems with the data. Some of the key concerns that I found:

- Street names are not consistent. Over-explanatory street names in some cases, for e.g. Tumkur Road, Next to Vijaya Vittala Temple or sometimes they're incomplete such as 17th Cross instead of 17th Cross road.
    - Examples of street name cleaning:
    - 17th Cross -> 17th Cross Road
    - Sarjapur rd -> Sarjapur Road
    - 12th Main, 6th Cross -> 12th Main, 6th Cross Road
- There are some weird values as far as some 'v' attribute values are concerned for tag attributes for e.g. {'k': 'name:kn', 'v': u'\u0c95\u0cc6.\u0c9a\u0ca8\u0ccd\u0ca8\u0cb8\u0c82\u0ca6\u0ccd\u0cb0'}. Since these are Unicode characters, they could be in some other language. On inspecting the entire element, I didn't find any key/value pair that would've indicated non-english language. Hence, there was no way to tell which language was that, so I've decided to include these tags but am not analysing them.

On checking for other errors, all the postcodes seem to be correct i.e. 6 digit numbers starting from '56'. Postcode type can be retained in string format itself in the final data model as no arithmetic operation is expected to be performed here on postcodes. There are some formatting issues in postal codes such as '560003 and 560 003'. These need to be rectified. Also, all postal codes need to be stripped of all white spaces from the beginning and the end. Further a regex check with r'560\d\d\d' has been run to identify postcodes that are not from

Bangalore urban that was the selected area. A few of the postcodes are from rural Bangalore or Odisa and they've been renamed to 'Fixme: Incorrect Value'.

- There are lot of new fields in several tag attributes and there is little consistency as one goes from element to element. To resolve this, I propose a data wrangling methodology as explained later in the section.
- On closer inspection, there are some 'k' attributes that are shouting for closer inspection. For example, 'FIXME' and 'fixme' records. These records will be ignored and the corresponding tags will be ignored.

Cleaning Street Names

- First step of the cleaning exercise is to segregate all the expected street names from the problematic/doubtful ones. That is accomplished by using the expected list as shown in code. All the street names not in expected list, are then included as a dictionary item to get clarity on both the street name and the problem associated.
- As a next step, discretionary decision needs to be made. Some items in the prob_street_names dictionary can be classified as acceptable street names after some modification. Let's discuss a few modifications now that will change the street names and make them acceptable.
    - If the street name ends with 'cross' or 'main', then we can append a Road to the end of the street name and it'll cease to be a problematic street name
    - Quite a few string names are very interestingly verbose or detailed out. There is a name of the road, but it also has additional information for e.g. 'Outer ring road, Bellandur'. In this case, I want to retain the street name as there is nothing wrong with the same. I also don't want to strip the information after 'road', as it provides useful additional information.
    - There are some street names such as '22nd Cross, HSR Layout, Sector 2' where addition of a string 'road' will make the street name acceptable. It also looks very logical because just an inconsistent naming convention shouldn't make us exclude an otherwise perfectly acceptable record entry.
    - On inspection, I also see that some street names have the abbreviation rd or Rd instead of Road. These outliers are also searched for and rd/Rd is changed to Road.
    - For all the rest of the street names, they've been replaced with a 'Fixme' warning sign so that users can assign the correct street name to the tags. This will be helpful to correct the street names and to make sure that future analysts are aware that they're incorrect.

Correcting Postcodes formatting errors

- Out of a total 960 postcodes in the smaller sample file, what we did was identified which entries were more than 6 in length. These were problem items and on closer inspections, there were formatting errors in the form of extra spaces. These extra spaces were removed using string.remove () function.
- Further a regex match is done to identify postcodes that don't belong to Bangalore urban. There are 10-15 odd values which are from Bangalore rural or from neighbouring states. By and large, data is correctly plotted for Bangalore urban. These outlier values are renamed to Fixme: Incorrect value

Data Shaping and Data Model Creation

Now that we've got all the helper functions to identify rogue tag values (e.g. FIXME), clean street names based on my domain knowledge of Bengaluru as a region, and correct the formatting of postal codes & removing gibberish tag attribute values, data wrangling is nearing completion and we can put this into a data model. To understand what kind of a data model we're talking about, we might want to look at the tag and associated data structure

with it to give us a clear idea of how to structure the data. One of the helper functions to do that is the tags collection dictionary that gives us an idea of the types of tags that are abundant in this xml file. From the OSM XML documentation, we know that node, way and relation are three data primitives represented by top-level tags here in the file. Let's first look at the desired data model for the same.

When it is a node, we can look at the following data model:

{

"id": "2406124091",

"type: "node",

"visible":"true",

"created": {

"version":"2",

"changeset":"17206049",

"timestamp":"2013-08-03T16:43:42Z",

"user":"linuxUser16",

"uid":"1219059"

},

"pos": [41.9757030, -87.6921867],

"address": {

"housenumber": "5157",

"postcode": "60625",

"street": "North Lincoln Ave"

},

"amenity": "restaurant",

"cuisine": "mexican",

"name": "La Cabana De Don Luis",

"phone": "1 (773)-271-5176"

}

This implies that the output will be a list of dictionaries. So, whatever be the original shape of data with the associated node tags, I need to, with the help of my helper functions, shape the data into this form for consumption into database.

Additionally for way, specifically we should do the following:

for "way" specifically:

<nd ref="305896090"/>

<nd ref="1719825889"/>

should be turned into

"node_refs": ["305896090", "1719825889"]

For relation, we should look at the following model:

{

"id": "1332095",

"type: "relation",

"created": {

"version":"1",

"changeset":"6717067",

"timestamp":"2010-12-20T15:35:51Z",

"user":"Alexander Hunziker",

"uid":"21825"

},

"ref_info": {

"ref": ["90631804",90631813]

"role": ["outer",inner]

"type": ["way", "way"]

},

"type": "multipolygon"

}

Import into Mongodb

From the command line, we use the mongo import command to import the json file output into a database to run some statistics on the data imported.

Running some queries:

1. Running query on data primitives in the database collection:
    a. File Sizes: Bengaluru_sample.osm: 126 mb
    b. Bengaluru_sample_osm.json: 196 mb
    c. Number of nodes is 568,585
    d. Number of ways is 130,529
    e. Code example: db.project.find({"type": "node"}).count()
    f. db.project.find({"type": "way"}).count()

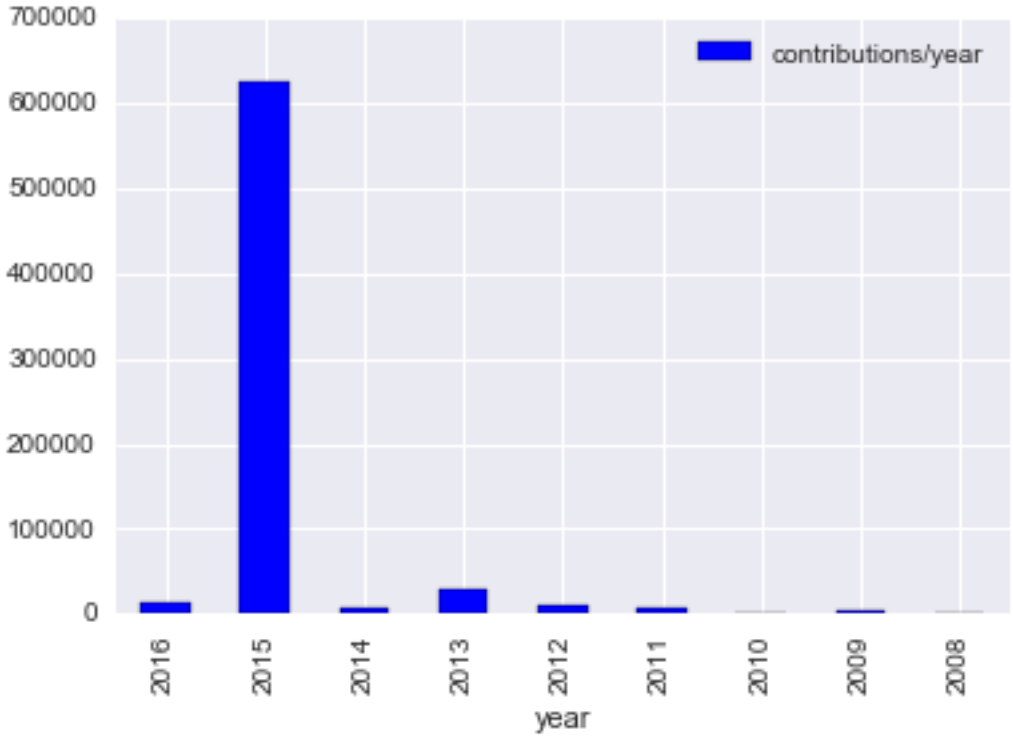
2. Who is the most active user contributing here in the data set?
    a. most_active_user_list = db.project.aggregate([
    b. {"$group": {"_id": "$created.user",
    c. "count" : {"$sum" : 1}}},

d. {"$sort": {"count": -1}},
e. {"$limit" : 20}])
f. for item in most_active_user_list:
g. pprint.pprint(item)
h. {u'_id': u'jasvinderkaur', u'count': 25228}. This is obtained by running an aggregation query, first grouping my most frequently occurring user names and then sorting it in descending order.
    i. {u'_id': u'jasvinderkaur', u'count': 25228}
    ii. {u'_id': u'akhilsai', u'count': 23909}
    iii. {u'_id': u'premkumar', u'count': 23260}
    iv. {u'_id': u'saikumar', u'count': 23133}
    v. {u'_id': u'shekarn', u'count': 19932}
    vi. {u'_id': u'vamshikrishna', u'count': 18897}
    vii. {u'_id': u'PlaneMad', u'count': 18245}
    viii. {u'_id': u'himalay', u'count': 17690}
    ix. {u'_id': u'himabindhu', u'count': 17455}
    x. {u'_id': u'sdivya', u'count': 16980}
    xi. {u'_id': u'hareesh11', u'count': 16519}
    xii. {u'_id': u'vamshiN', u'count': 16217}
    xiii. {u'_id': u'harishk', u'count': 15395}
    xiv. {u'_id': u'sampath reddy', u'count': 14379}
    xv. {u'_id': u'bindhu', u'count': 13996}
    xvi. {u'_id': u'kranthikumar', u'count': 13677}
    xvii. {u'_id': u'Navaneetha', u'count': 13492}
i. How have contributions changed over time to the project?
    i. contribution_16 = db.project.find({ "created.timestamp": { "$regex": "2016" } }).count()
    ii. contribution_15 = db.project.find({ "created.timestamp": { "$regex": "2015" } }).count()
    iii. contribution_14 = db.project.find({ "created.timestamp": { "$regex": "2014" } }).count()
    iv. contribution_13 = db.project.find({ "created.timestamp": { "$regex": "2013" } }).count()
    v. contribution_12 = db.project.find({ "created.timestamp": { "$regex": "2012" } }).count()
    vi. contribution_11 = db.project.find({ "created.timestamp": { "$regex": "2011" } }).count()
    vii. contribution_10 = db.project.find({ "created.timestamp": { "$regex": "2010" } }).count()
    viii. contribution_09 = db.project.find({ "created.timestamp": { "$regex": "2009" } }).count()
    ix. contribution_08 = db.project.find({ "created.timestamp": { "$regex": "2008" } }).count()
    x. map_contribution_per_year = [contribution_16, contribution_15, contribution_14, contribution_13, contribution_12, contribution_11, \
    xi. contribution_10, contribution_09,contribution_08]
    xii. year = [2016, 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008]
    xiii. contribution_dataset = list(zip(map_contribution_per_year, year))
    xiv. contribution_df = pd.DataFrame(data = contribution_dataset, columns=['contributions/year', 'year'])
    xv. contribution_df
    xvi. The most prolific year for making contributions on the map was 2015 with 625000 contributions and the least prolific is 2008 with 1466

contributions. We use $regex operator to query and find the count of these records. Later on, the data can be visualized on the bar graph to see for yearly movement in contributions.

xvii. Below are the contributions made to the OSM data on a yearly basis:

xviii. 0 135742016
xix. 1 625477 2015
xx. 2 7038 2014
xxi. 3 279932013
xxii. 4 9981 2012
xxiii. 5 6576 2011
xxiv. 6 1790 2010
xxv. 7 5409 2009
xxvi. 8 1466 2008



j.

3. Additional Aggregation analytics:
   a. pipeline = [{"$group": {"_id": "$address.city",
   b. "count": {"$sum" : 1}}},
   c. {"$sort": {"count": -1}}]
   d. for item in db.project.aggregate(pipeline):
   e. pprint.pprint(item)
   f. Bangalore as a city is mostly not mentioned in most records. This indicates that records are incomplete. Out of a total of 699310 records, 698993 have 'None' as city value or don't have that field at all.
   g. pipeline = [{"$group": {"_id": "$amenity",
   h. "count": {"$sum" : 1}}},
   i. {"$sort": {"count": -1}}]
   j. for item in db.project.aggregate(pipeline):
   k. pprint.pprint(item)
   l. On looking at amenity statistics, what we observe that there are 247 counts of restaurant, 183 counts of places of worship, 121 schools and 89 hospitals.
      i. {u'_id': u'restaurant', u'count': 247}
      ii. {u'_id': u'place_of_worship', u'count': 183}
      iii. {u'_id': u'atm', u'count': 130}

iv. {u'_id': u'bank', u'count': 129}
v. {u'_id': u'school', u'count': 121}
vi. {u'_id': u'hospital', u'count': 89}
vii. {u'_id': u'fast_food', u'count': 81}
viii. {u'_id': u'college', u'count': 72}
ix. {u'_id': u'fuel', u'count': 67}
x. {u'_id': u'pharmacy', u'count': 60}

m. pipeline = [{"$match" : {"amenity" : "restaurant"}},
n. {"$group": {"_id": "$cuisine",
o. "count": {"$sum" : 1}}},
p. {"$sort": {"count": -1}}]
q. for item in db.project.aggregate(pipeline):
r. pprint.pprint(item)
s. Amongst the restaurant amenities, the top cuisine is 'Indian' with 49 entries.
t. pipeline = [{"$match" : {"amenity" : "place_of_worship"}},
u. {"$group": {"_id": "$religion",
v. "count": {"$sum" : 1}}},
w. {"$sort": {"count": -1}}]
x. for item in db.project.aggregate(pipeline):
y. pprint.pprint(item)
z. The top religion in the city is dominated by Hindus. This is evident from the largest count of places of worship belonging to 'hindu' with count of 108 out of 183 places of worship.
i. {u'_id': u'hindu', u'count': 108}
ii. {u'_id': u'christian', u'count': 30}
iii. {u'_id': None, u'count': 25}
iv. {u'_id': u'muslim', u'count': 18}
v. {u'_id': u'hinduism', u'count': 1}
vi. {u'_id': u'sikh', u'count': 1}

aa. pipeline = [{"$match" : {"amenity" : "atm"}},
bb. {"$group": {"_id": "$name",
cc. "count": {"$sum" : 1}}},
dd. {"$sort": {"count": -1}}]
ee. for item in db.project.aggregate(pipeline):
ff. pprint.pprint(item)
gg. I tried investigating the largest network of atms in the city to find out the most widely present bank but most of the information is incomplete regarding the same with 'None' accounting for the majority of them.
i. {u'_id': None, u'count': 88}
ii. {u'_id': u'Canara Bank', u'count': 3}
iii. {u'_id': u'Axis Bank ATM', u'count': 3}
iv. {u'_id': u'Canara Bank ATM', u'count': 2}
v. {u'_id': u'SBI ATM', u'count': 2}
vi. {u'_id': u'ICICI Bank', u'count': 2}
vii. {u'_id': u'ICICI', u'count': 2}

Other ideas about the dataset

One thing that I notice is that while data contribution is distributed rather uniformly across top 20 users. That said, the data contribution is rather incomplete. Most contributors add fields that are different from one another which results in making it difficult to define a uniform data model for analysis later. Maybe some kind of gamification strategy or stringent

data contribution policy can be implemented to collect and report data points in a pre-defined manner.

Conclusion

After this review of the data, it's obvious that the Bengaluru/Bangalore data is incomplete with a lot of missing fields. These missing fields are a hindrance in running analytics on the dataset as the insights are very limited in scope. Also, there are a large number of fields that may or mayn't have a lot of significance in building insights. Some kind of a more robust data model needs to be established to investigate this data set in a more efficient manner and to draw more significant insights. Additionally, what'll also help is cleaner, more complete data to fit into this data model for us to analyse this data set.

Benefits of a more robust data model:

1. Standardization of fields for different data primitives and sub structure types
2. Ensures more fields being filled with real values that 'none'. Right now, as we can see that in most of our queries, the top result is the 'None' field which indicates that data isn't available to draw much insights. This could go a long way in helping us draw tangible analysis points from this data.

Potential Problems

The above solution of introduction standardization and more robustness will have an impact on the flexibility of this task before. As we have seen earlier, there are quite a few users making contribution to the data and it is likely that they enjoy the freedom in reporting the data as they see it. Adhering to a 30,000 feet policy on how to contribute to data could jeopardize the process and some may feel that there is a force fitting that happens instead of raw data being reported.

This could lead in loss of motivation as some kind of control is built-in the method of data collection and reporting and that mayn't go down too favourably. This might see a reduction in data contribution but the good part is that it is likely to be offset by more complete, better quality data.