Udacity
OpenStreetMap project

Data Wrangling with MongoDB

City of Philadelphia, Pennsylvania.

—

Anirudh Ramesh

(https://mapzen.com/data/metro-extracts/metro/philadelphia_pennsylvania/)

1. Cleaning the Map data.
2. Overview of the data
3. Additional Suggestions/Ideas.
4. Conclusion

Cleaning the Map data :

There were some outliers I had to examine during the cleaning stage. I think most of them fit into the following three patterns.

1. Simplification (or abbreviation) of names without any legend or standards.
   Ex : 1. Ave, Ave. , ave. for Avenue.
   2. ter, Ter, Ter. For terrace etc.
   Solution :
   On looking through the document manually, I found most of the abbreviated words were usually the last words of a 'name' key tag. i.e. Lane, Street, Building etc.. Set of all last words of a name.
   I used the same logic involving regular expression and word boundaries from the lesson to solve this problem. (Refer function : update_name in 'audit_v1.py'.)

2. Redundant or unnecessary key-value pairs.
   For example , a building maybe classified as ‘memorial’ - but the same is expressed in multiple ways across the OSM file (for other tags also).
   <tag k = ‘memorial’ v = ‘yes’>
   <tag k = ‘memorial:type’ v = ‘statue’>
   <tag k = ‘historic’ v = ‘Memorial - Statue of xxx’>
   Solution :
   I queried all the value provided by each key and decided on a set of key that I have to ignore either because - they were tags, which would not help me during analysis(for instance - ‘FIXME’) or is redundant similar the given example. (Memorial is in the list key_set_ignore in the audit_v1.py for this reason). Some of the Key-Value pairs also did not follow convention as mentioned in OSM wiki.

3. Wrongful hierarchy/grouping of nodes.
   For example , The key tag i**s_in** is a separate tag even though it can be a part of address tag (GNIS or TIGER) . The tag ‘survey’ gives the survey date , that needs to be manually extracted , similar to how census tag gives both ‘population’ and ‘year’ of census in a single k-v tag.

I have tried to stick to OSM wiki for classification for the most part. However, I had to change a few keys - for instance , ‘created_by’ to ‘source’, since , I wanted a ‘created_by’ tag to display information of the user who had created the data. I have also re-classified some of the nodes, while it still follows the guidelines of OSM , to make it more readable. Any changes I’ve done are commented in the code (‘audit_v1.py’)

Overview of the data :

1. **Size of the OSM FIle :** philadelphia_pennsylvania.osm = 592 MB.
2. **Size of the JSON File :** output.json = 836 MB.
3. **Number of documents :** > db.data.find().count()
   3061653
4. **Number of way tags :** > db.data.find({'tag_type' : 'way'}).count()
   259444
5. **Number of node tags :** > db.data.find({'tag_type' : 'node'}).count()
   2802209

6. **Number of unique users :** > db.data.distinct('created_by.user').length
   1811
7. **Number of data taken from each source :**

1. TIGER → > db.data.find({'tiger' : {'$exists' : true}}).count()
   89606
2. National Hydrography dataset : > db.data.find({'nhd' : {'$exists' : true }}).count()
   20393
3. Office of geographic information (MASSGIS) → > db.data.find({'massgis' : {'$exists' : true}}).count()
   4

4. US Board of geographic names (GNIS) → > db.data.find({'gnis' : {'$exists' : true }}).count()
   6000

8. Number of places for leisure activities : > db.data.find({'leisure' : {'$exists' : true}}).count()
4787

9. Most common amenities : > db.data.aggregate([{"$match":{"amenity":{"$exists":1}}}, {"$group":{"_id":"$amenity", "count":{"$sum":1}}}, {"$sort":{"count" : -1}}, {"$limit": 5}])

- {"_id" : "parking", "count" : 5131 }
- {"_id" : "school", "count" : 2139 }
- {"_id" : "restaurant", "count" : 1200 }
- {"_id" : "place of worship", "count" : 1090 }
- {"_id" : "fire station", "count" : 513 }

Additional Ideas:

I notice , There are a total of 2802209 nodes (points) on the map.

> db.data.find({'tag_type' : 'node' , 'source' : {'$exists' : false}}).count()
2598171.

This is the amount of nodes in OSM , without imports from external sources. A majority of

this entry , is from various bots. One such example is woodpeck_fixbot , which is a bot written by user woodpeck . There are several such bots('NE2','bot-mode' to name a few).

> db.data.find({'tag_type' : 'node' , 'created_by.user' : 'woodpeck_fixbot' }).count()
572034.

In other words, 22% of data from external sources , is from a single bot - woodpeck_fixbot.
Total documents created by various bots :

> db.data.find({'created_by.user' : /bot/}).count()

617245

Out of 204038 documents , 617245 are created by bots.
To add to this information ,

> db.data.find({'tiger.reviewed' : 'no'}).count() → 79014

> db.data.find({'tiger.reviewed' : {'$exists' : true}}).count() → 79598

Out of all the documents that have been included in 'way' tag , where ways are chosen from TIGER, only 584 have been verfied- presumably because, these were human inputs. Therefore, a large portion of imported nodes are **unverified** and a big chunk of data in the map is from **bots/scripts.**

Also, 204038 are imported from various sources. I.e these are the nodes which **were not filled by bots** but the information was not available first-hand also ;, the top 5 sources are :

> db.data.aggregate([{'$match' : {'source' : {'$exists' : true}}} , {'$group' :

{'_id' : '$source' , 'count' : {'$sum' : 1}}} , {'$sort' : {'count' : -1 }} , {

'$limit' : 5}])

{ "_id" : "ArcGIS Exporter", "count" : 197071 }

{ "_id" : "NHD import v0.4 20110710211953", "count" : 13823 }

{ "_id" : "NJ2002LULC", "count" : 11217 }

{ "_id" : "Bing", "count" : 6035 }

{ "_id" : "gnis", "count" : 6000 }

According to the wiki - http://www.pasda.psu.edu/ is the preffered link for accesing geo-spatial data. But , as we can see, there is a large use of third party services like ArcGIS and even third-party maps like bing - not that it will necessarily make the map more/less accurate or correct. Not only is the data un-verified, but it is also from a plethora of sources.

The amount of openstreetmap users, using 'JOSM' is :

```
> db.data.find({'source' : 'JOSM'}).count()
  281
```

Or, Only 281 entries which were not available in the map , were actually inputs from various users.

Solution : If only 281 entries are from users, it is safe to assume very low level of awareness of OSM. Thus , it is necessary to create a bigger user-base. This will lead to more accurate and detailed entries , and will reduce unverified data. Another important step is to verify data entered from various sources. This can also help the product more usable, automatically increasing the user-base.

There are also downsides to working with a larger user base. Verifying the data , although increases the accuracy of the map, this process will result in a huge dataset. The data can be untrustworthy , or there might be outliers or data which gives completely opposite information compared to the one available. This might also lead to a level of granularity that is unnecessary or the redundancy of data that is hard to solve.
(This is more elaborate in the source_links file - link 1).

Conclusion :

From my audits, I find most of the data in 'philadelphia_pennsylvania.osm' follow the conventions, and there is no scarcity of data. But the user-base looks to be really small and not all of the third-party imported data is verified (although they may potentially be correct). I feel making these changes would improve the OSM of the Philadelphia city. This analysis is not however, complete - since the data set is very large.