P3- Data Wrangling with Mongo DB

(Open Street Map)

Map Area: New York City, United States of America
http://www.openstreetmap.org/relation/175905#map=10/40.6978/-73.9792

Contents

1. Problems encountered in the map

After downloading a custom extract of the NY City area and running a sample data set against my process_data.py file, I noticed the following problems with the data:

1.1 Problem in the extract

While obtaining the smaller sample, there was some problem in the dataset, due to which I could not parse the extract to obtain a smaller data set. On investigating the source of error, I found invalid characters like below were present in the custom extract:

<tag k="&#2;" v="Blunauer"/> line no 1515617

Since it was creating issues in parsing the file, I had to manually remove such lines from the dataset.

1.2 Presence of Non ASCII characters in the data set:

While processing the data set, I found non ASCII characters/other language symbols found in the dataset like :

<tag k="name" v="慈濟大學"/>
<tag k="name:zh" v="慈濟大學"/>
<tag k="addr:city" v="花蓮縣"/>
<tag k="addr:full" v="97004花蓮縣花蓮市中央路三段701號"/>
<tag k="wikipedia" v="zh:慈濟大學"/>
<tag k="addr:street" v="中央路三段"/>
<tag k="addr:district" v="花蓮市"/>
</way>

Since it was not possible to have Chinese/any other language/non ASCII values for any of the key/value pair in the New York city dataset, I skipped the processing of such characters in my python function is_ascii in my process_data.py file:

```
def is_ascii(str):
        return all(ord(chars) < 128 for chars in str)
```

1.3 Street and city name abbreviations

On analyzing the dataset using my audit_sample.py python file, I found the street and city name in the dataset being abbreviated. I I updated all substrings in problematic address strings, such that "W 56th St #1H" becomes "West 56th Street #1H" using update_name() in my process_data.py.

1.4 Incorrect postal codes

I executed the below query to find the postal codes occurring in the data set with their frequency:

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

Here are the top 3 results with count:

```
{ "_id" : "11101", "count" : 1739 }
{ "_id" : "11211", "count" : 1723 }
{ "_id" : "11206", "count" : 1575 }
        ….
```

On analyzing the data I found that many of the postal codes contained alphabets and did not conform to the standard zip code format of 5 numeric digits. So I cleaned the data for these invalid pin codes in my data set in my process_data.py.

1.5 Incorrect city names

I executed the below query to find the city names occurring in the data set with their frequency:

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

Results for the corresponding cities

```
{ "_id" : "New York", "count" : 1997 }
{ "_id" : "?????????", "count" : 835 }
{ "_id" : "Imus", "count" : 792 }
{ "_id" : "Brooklyn", "count" : 707 }
……
```

On analyzing these results, I found certain incorrect city names like "?????????" were found in the result and also I found many of the cities returned in the result were not a part of the United States itself like Imus in in Philippines, Anderlecht in Belgium and Gundersheim in Germany.

Second level tags whose first level tags were already encountered in the extract

```
<node changeset="24427052" id="2986320002" lat="40.6783625" lon="-73.9481562"
timestamp="2014-07-29T17:41:45Z" uid="1851008" user="YamaOfParadise" version="1">
        <tag k="railway" v="stop" />
        <tag k="operator" v="Long Island Rail Road" />
        <tag k="public_transport" v="stop_position" />
        <tag k="railway:position" v="1.6" />
</node>
```

Here tag k="railway" was already encountered in the extract and then k="railway:position" was again found. I created a dictionary for railway in this case and put the corresponding key in it like the below json using my process_data.py :

{"name": "Nostrand Avenue", "created": {"changeset": "24427052", "user": "YamaOfParadise", "version": "1", "uid": "1851008", "timestamp": "2014-07-29T17:41:45Z"}, "pos": [40.6783625, -73.9481562], "train": "yes", "public_transport": "stop_position", "operator": "Long Island Rail Road", "railway": {"position": "1.6", "railway": "stop"}, "type": "node", "id": "2986320002", "network": "Long Island Rail Road"}

Other second level tags

Address related information in the second level "k" tags pulled from Tiger data were also divided into segments like :

```
<tag k="tiger:cfcc" v="A63" />
<tag k="tiger:county" v="New York, NY" />
<tag k="tiger:reviewed" v="no" />
```

These have also been split in a second level dictionary like the address field via my process_data.py.

2. Overview of the Data

Here in this section I provide some general information about the dataset like file sizes and compute statistical counts using Mongo DB queries. Also I have computed some additional statistics about the dataset as well.

Statistical overview

File sizes:

new_york_city.osm	258.84 MB
new_york_city.osm.json	288.88 MB

Here the new_york_city.osm is the custom extract file downloaded from Open Street map and new_york_city.osm.json is the json file generated from prepare_sample.py which is pushed into collection "nyc" using mongoimport.

Size of database

```
> db.nyc.dataSize()
322945653
```

Number of records:

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

1316799

Number of unique users :

```
> db.nyc.distinct("created.user").length
11640
```

Number of nodes :

```
> db.nyc.find({“type”:”node”}).count()
1096579
```

Number of ways :

```
> db.nyc.find({"type": "way"}).count()
219988
```

Contribution of “bot” users

```
> db.nyc.find({"created.user": {$in:[/^bot/,/bot$/]}}).count()
7203
```

Here I have assumed that “bot” users have name beginning with “bot” or ending with “bot”.

Proportion of “bot” user contribution

```
>  db.nyc.aggregate([ { $project:{_id: 0, bot_proportion:{$divide: [ db.nyc.find({"created.user":
{$in:[/^bot/,/bot$/]}}).count(),db.nyc.find({"created.user":{"$exists":1}}).count()]}}},{"$limit":1}])

        { "bot_proportion" : 0.0054700831334167175 }
```

Here we can see that the percentage of bots contributing to the dataset is very less i.e. 0.54% as compared to the real users contribution.

Number of roads in the city

```
> db.nyc.find({"highway":{$in:["motorway","trunk","primary","secondary","tertiary"]}}).count()
7084
```

Number of eating joints in the area

```
>  db.nyc.find({"amenity":{$in:["cafe","fast_food","restaurant","food_court","biergarten"]}}).
count()
2541
```

Number of vegan cafes

```
> db.nyc.aggregate([{"$match":{"diet.vegan":{"$exists": true,"$ne": null},"amenity": "cafe"}},
{"$group": {"_id":"$diet.vegan", "count":{"$sum":1}}}])

        { "_id" : "yes", "count" : 40 }
```

This shows that there are about 40 vegan cafes in this dataset

Number of educational institutes

```
> db.nyc.find({"amenity":{$in:["college","kindergarten","school","university"]}}).count()
694
```

Number of financial institutions

```
> db.nyc.find({"amenity":{$in:["atm","bank","bureau_de_change"]}}).count()
346
```

Number of medical help centers available

```
> db.nyc.find({"amenity":{$in:["clinic","dentist","doctors","hospital","nursing_home",
"pharmacy"]}}). count()
296
```

Number of recreational centers

```
> db.nyc.find({"amenity":{$in:["cinema","theatre","arts_centre","community_centre"]}})
.count()
123
```

Number of Starbucks in NYC

```
> db.nyc.find({"amenity":"cafe","name":"Starbucks"}).count()
89
```

3. Other ideas about the datasets

Using Mongo DB I explored the data set further using the following queries :

Most contributing real users:

```
> db.nyc.aggregate({$match:{"created.user": {$nin:[/^bot/,/bot$/]}}},
{"$group":{"_id":"$created.user","count":{"$sum":1}}},  {"$sort":{"count":-1}},{"$limit":5})
```

```
{ "_id" : "Rub21_nycbuildings", "count" : 411984 }
{ "_id" : "smlevine", "count" : 156928 }
{ "_id" : "lxbarth_nycbuildings", "count" : 79541 }
.....
```

Most contributing "bots"

```
> db.nyc.aggregate({$match:{"created.user": {$in:[/^bot/,/bot$/]}}},
{"$group":{"_id":"$created.user","count":{"$sum":1}}}, {"$sort":{"count":-1}},{"$limit":5})
```

```
{ "_id" : "woodpeck_fixbot", "count" : 5100 }
{ "_id" : "botdidier2020", "count" : 1556 }
{ "_id" : "bot-mode", "count" : 350 }
.....
```

Top amenities

```
> db.nyc.aggregate([{"$match":{"amenity":{"$exists":1}}}, {"$group":{"_id":"$amenity",
"count":{"$sum":1}}}, {"$sort":{"count":-1}},{"$limit":10}])
```

```
{ "_id" : "bicycle_parking", "count" : 3528 }
{ "_id" : "restaurant", "count" : 1562 }
{ "_id" : "parking", "count" : 1146 }
...
```

As observed in the result, the top amenities were bicycle parking, restaurants and parkings

Most popular capacity of bicycle parkings

```
> db.nyc.aggregate([{"$match":{"capacity":{"$exists": true,"$ne": null}, "amenity":
"bicycle_parking"}},{"$group": {"_id":"$capacity", "count":{"$sum":1}}},  {"$sort":{"count":-1}},
{"$limit": 3}])
```

```
{ "_id" : "2", "count" : 2131 }
{ "_id" : "5", "count" : 545 }
{ "_id" : "capacity", "count" : 275 }
```

Here the maximum capacity had been for 2.

Types of parking

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

```
{ "_id" : "surface", "count" : 389 }
{ "_id" : "underground", "count" : 40 }
{ "_id" : "multi-storey", "count" : 19 }
{ "_id" : "garage", "count" : 5 }
{ "_id" : "lane", "count" : 1 }
```

These results show that surface level parking is most common in NYC, followed by underground, multi storey, garage and lane

Most common cuisine in cafes

```
> db.nyc.aggregate([{"$match":{"cuisine":{"$exists": true,"$ne": null},"amenity": "cafe"}},
{"$group": {"_id":"$cuisine", "count":{"$sum":1}}}, {"$sort":{"count":-1}} ,{$limit:5}])
```

```
{ "_id" : "coffee_shop", "count" : 163 }
{ "_id" : "donut", "count" : 14 }
{ "_id" : "sandwich", "count" : 7 }
....
```

Shopping places near 5th Avenue

```
> db.nyc.find( { pos : { $near : { $geometry : { type : "Point" , coordinates : [40.7314123,-
73.9991735] }, $maxDistance: 5000} }, "shop": {$in:["clothes","fashion"]}} ,
{_id:0,name:1,address:1})
```

Below are some of the shopping places from a list of 261.

```
{ "name" : "Prada" }
{ "name" : "H&M" }
{ "name" : "Saint Laurent Paris", "address" : { "housenumber" : "80" } }
{ "name" : { "en" : "Victoria's Secret", "name" : "Victoria's Secret" } }
.....
```

Eating joints near Madison Square Garden

```
> db.nyc.find({ pos : { $near :  { $geometry : {type : "Point" , coordinates : [40.7505045,-
73.9956274] },$maxDistance: 5000} }, "amenity": {$in:["cafe","restaurant"]}} ,
{_id:0,name:1,address:1,cuisine:1})
```

Below are some of the eating joints from a list of 1843

{ "cuisine" : "indian", "name" : "Banana Leaf" }
{ "cuisine" : "italian", "name" : "Bella Napoli", "address" : { "city" : "New York", "state" : "NY", "housenumber" : "257", "postcode" : "10001" } }
{ "cuisine" : "mediterranean", "name" : "Hell's Kitchen", "address" : { "city" : "New York City", "housenumber" : "523", "postcode" : "10036" } }

4. Improvements in the dataset

```
> db.nyc.find({name:{$exists:false}}).count()
    1288524

> db.nyc.find({pos:{$exists:false}}).count()
    220069
```

On analyzing the results executed by the above queries in Mongo DB, I find many of the nodes/ways tags missing their name and geospatial information which makes it difficult to analyze lot of information present in the dataset. Also many of the common amenity related information seems to be missing as it is difficult to believe that the number of financial institutions, educational institutes and medical help centers would be less than the number of eating joints in the city! For the most popular and populous city in the US, these results seems to be inconsistent.

```
> db.nyc.find({"leisure":"park","name":"Central Park"}).count()
    0
```

Also “Central Park” being the most popular park in NYC, is found missing in the dataset. So I think more users should be encouraged more to contribute geospatial information about their neighborhood to improve this dataset. Also probably an integration with other 3rd party Location based APIs like Foursquare would help in leaps and bounds to improve the quality of data in Open Street Map.

5. Conclusion

After inspecting my dataset, I would like to conclude that Open Street map is a great open source initiative to access geo-spatial information about any place in the world without being physically present in that area of the world. Though it has its share of human generated errors like the presence of presence of non – interpretable characters and other non ASCII characters in the data set, incomplete and inconsistent information like the presence of other countries information in NYC dataset. As part of my Data Wrangling using Open Street Map data project, I have made an effort to clean such incorrect data using process_data.py. Since it’s not possible to be fool proof in any GIS related information, probably better initiatives for users to promote contribution to the dataset as well as integration with other 3rd party API’s and cleaning data using similar scripts can go a long way in improving the quality of the datasets in the Open Street Map.