Edward Farrar
edward.farrar@ge.com
Due: 29 June 2015
Udacity Data Analyst Nanodegree

Data Wrangle OpenStreetMap Data

Data Wrangling with MongoDB

Map Area: Fern Creek/Highview suburbs of Louisville, KY
https://www.openstreetmap.org/export#map=13/38.1443/-85.6040
http://overpass-api.de/api/map?bbox=-85.6824,38.1058,-85.5361,38.1719

Problems Encountered in the Map

Initially, I wanted to wrangle the entire Louisville, KY metro area. Map Zen had a metro download for Louisville. When extracted, the resulting data totaled just over 1GB. I started with a basic Python script called AuditMapData that counted the tags in order to get an idea of how many nodes and ways were included. However, I quickly learned that processing even a relatively small 1GB file was more than my small home computer could handle. The Python Notebook froze and system memory hit 100% for about 20 minutes.

So, I realized I either needed a more powerful system or a smaller data set. While a SPARC T5-2 from work would have been nice to use, I opted to shrink the data set to the Fern Creek/Highview suburbs of Louisville, KY – my neighborhood and surrounding area. This data set was 86.1MB and could easily be processed by my computer in about a minute or two.

Once the data set size was taken care of, I started expanding the AuditMapData code. I audited street names and much to my surprise, there were very few problems. In fact, I only found 2 types that needed to be converted from abbreviations: Rd (Road) and PT (Point), and these were only present on total of 12 out of over 33,000 streets. I then started checking other things like zip codes, city names, and state names. All were consistent and accurate for this area. I then branched out to amenities – again all look consistent.

At this point, I was getting desperate to find something to correct in the data! I decided to look at combinations of amenity and building. I then found that there was an inconsistency in how some pairs were represented. Specifically, I found that for “places_of_worship”, building was sometimes “church” and sometimes “yes”. Similarly, for “fuel”, the building was sometimes “roof” and sometimes “yes”. After looking at the counts for these “buildings” in relation to the amenity, I found that the consistent value was “yes”.

At this point, I created a data cleaning plan to correct these three data inconsistency issues:

- Rd (Road) and PT (Point) street name abbreviations: simply identified the addr:street values ending in Rd or PT and made the appropriate substitution.
- “place_of_worship” and “building” combination: after reviewing the audit data, I decided to set the building to “yes” for every amenity “place_of_worship” in order to make all entries consistent.

- "fuel" and "building" combination: after reviewing the audit data, I decided to set the building to "yes" for every amenity "fuel" in order to make all entries consistent.

After creating the plan, I created a Python script called Clean and Convert MapData to implement the data corrections, format the data, and create the json file.

Data Overview

FernCreek-Highview_Kentucky.osm: 86.1MB

FernCreek-Highview_Kentucky.osm.json: 124MB

I used mongoimport to create the osm database and the ferncreek collection with the generated json data.

Sample queries

```
C:\mongodb\bin>mongo

2015-06-22T19:32:40.662-0400 I CONTROL  Hotfix KB2731284 or later update is not
installed, will zero-out data files

MongoDB shell version: 3.0.2

connecting to: test

> use osm

switched to db osm
```

Number of documents in the database:

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

410893
```

Number of nodes:

```
> db.ferncreek.find({"type":"node"}).count()

370352
```

Number of ways:

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

40541
```

Number of distinct users:

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

69
```

Top contributor:

```
> db.ferncreek.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":1}])
```

```
{ "_id" : "Omnific_louisvilleimport", "count" : 194424 }
```

Number of users who only submitted one entry:

```
> db.ferncreek.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$group":{"_id":"$count", "num_users":{"$sum":1}}}, {"$sort":{"_id":1}},
{"$limit":1}])
{ "_id" : 1, "num_users" : 7 }
```

Number of amenities:

```
> db.ferncreek.aggregate([{"$group":{"_id":"$amenity", "count":{"$sum":1}}},
{"$sort":{"count":-1}}])
{ "_id" : null, "count" : 410777 }
{ "_id" : "place_of_worship", "count" : 33 }
{ "_id" : "parking", "count" : 26 }
{ "_id" : "school", "count" : 15 }
{ "_id" : "restaurant", "count" : 14 }
{ "_id" : "fast_food", "count" : 11 }
{ "_id" : "fuel", "count" : 5 }
{ "_id" : "pharmacy", "count" : 4 }
{ "_id" : "fire_station", "count" : 2 }
{ "_id" : "grave_yard", "count" : 2 }
{ "_id" : "post_box", "count" : 2 }
{ "_id" : "police", "count" : 1 }
{ "_id" : "post_office", "count" : 1 }
>
```

Additional Ideas

After further review of the top users, it's obvious that the majority of the data was imported from known, clean sources. One idea to verify the data might be to compare the data with USPS data. The USPS has standardized addresses for all mailboxes in the country. Cross-referencing the data could reveal missing or incorrect addresses in Open Street Maps.

Conclusion

While the map area I chose was small out of necessity, it allowed me to run additional queries to "validate" the data. For instance, looking at the counts for all shops in the area, shows this result:

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


{ "_id" : "department_store", "count" : 1 }
{ "_id" : "pet", "count" : 1 }
```

```
{ "_id" : "yes", "count" : 1 }
{ "_id" : "hardware", "count" : 1 }
{ "_id" : "hairdresser", "count" : 1 }
{ "_id" : "beauty", "count" : 1 }
{ "_id" : "car_repair", "count" : 1 }
{ "_id" : "supermarket", "count" : 4 }
{ "_id" : "convenience", "count" : 6 }
```

The “department_store” was interesting as I didn't think we had one. When I found the document, it was “Kohl's”, which was built just down the street from my neighborhood 2 years ago. I just didn't think of it being a department store. The shop recorded as “yes” turned out to be “JD Becker”, which is a specialty store that sells University of Louisville and University of Kentucky merchandise. All the counts seem reasonable. Spot checking the supermarkets, I found Valu Market, Kroger, Walmart, and Price Less Foods, which are all the groceries located within the bounding box I selected for this project.

Overall, given the cleanliness of the data, I would be concerned that future students may find it difficult to complete the assignment. While working with the data is good, artificial changes to the data may become necessary to demonstrate the concepts learned (i.e. changing all street names to capital letters, inserting 9 digit zip codes, etc.)