OpenStreetMap Sample Project
Data Wrangling with MongoDB
Ji Gu

Project Summary

What is your name?

Ji Gu

What E-mail address do you use to sign in to Udacity?

jigucci@gmail.com

What area of the world you used for your project? Post a link to the map position and write a short description. Note that the osm file of the map should be at least 50MB.

Map Area: Atlanta, GA, United States
URL:
http://metro.teczno.com/#atlanta
35MB bzip'ed XML OSM data: http://osm-extracted-metros.s3.amazonaws.com/atlanta.osm.bz2

I chose this particular place because it is my neighborhood, I know it well and would like its map to be improved in quality!

Is there a list of Web sites, books, forums, blog posts, github repositories etc that you referred to or used in this submission (Add N/A if you did not use such resources)?

Standard address abbreviations - http://www.semaphorecorp.com/cgi/abbrev.html
Georgia zip codes - http://www.mapsofworld.com/usa/zipcodes/georgia/

Please carefully read the following statement and include it in your email:

"I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc. By including this in my email, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate."

Is there any other important information that you would want your project evaluator to know?
Use this space to communicate with your project evaluator. Is there anything you would like to communicate? Feedback or suggestions?

Please note the dataset imported into MongoDB ONLY contains info from element.tag = "node" or element.tag = "way".

1. Problems Encountered in the Map

After initially downloading of the whole Atlanta area data and running it against a provisional data.py file, I noticed three main problems with the data, which I will discuss in the following order.

1. Non-standard postcode (e.g. "Atlanta,")
2. Incorrect city name
3. Issues with values in "address.street" element
    - incomplete street name
    - website is used as street name
    - incorrect street address abbreviation (refer to "Standard address abbreviations" in the project summary)
    - inconsistent direction naming in the street name

Non-standard postcode

Once the data was imported to MongoDB, some basic querying revealed several non-standard postcodes. Please note postcode format in both “28226-0783” and “28226” are assumed/believed to be standard format in the USA. Formats other than these two are non-standard postcodes in the USA.

By running Atlanta-zipcode.py on the raw atlanta.osm file, the following results can be retrieved.

```
{'other': 9, 'zipcode': 38294}
{'2034823412': '300313',
 '205002773': 'GA 30326',
 '238500269': 'GA 30092',
 '238500271': 'GA 30071',
 '238501188': 'GA 30092',
 '238917119': '1879',
 '2562004187': 'GA 30307',
 '367915530': '300313',
 '42882100': 'Atlanta,'}
```

#Number of correct postcode:
38294

#Number of incorrect/non-standard postcode:
9

By using some regex in Atlanta-zipcode.py, the incorrect postcodes can be identified and corrected by finding out the right ones. The four types of errors are shown below (Please note I use ‘id’ key to help identify the unique row.)

- postcode “Atlanta,” - for instance found in ‘id”=42882100, changed to 30350 based on lat/long.
- postcode “300313” - for instance found in “id”=367915530, changed to 30313
- postcode “GA 30092” - for instance found in “id”=238500269, changed to 30092
- postcode “1879” - found in “id”=238917119, changed to 30092 based on the location.

These obvious incorrect postcodes have to be fixed in order to maintain high quality of the dataset. A more in-depth look at the accuracy of the postcode against address and county is presented in the following ‘Additional Ideas’ section.

Incorrect city name

By running the following query, several wrong city names are found out. It seems most of them are related to typo.

#Number of city name appearing only once: (all query commands are in blue color)

```
db.atl2.aggregate([{"$match":{"address.city":{"$exists":1}}},{"$group":{"_id":"$address.city","count":{"$sum":1}}},{"$sort":{"count":1}},{"$limit":5}])
```

Format: Incorrect one => Correct one

Atlanda => Atlanta
Palmatto => Palmetto
GA 30350 => Atlanta (Georgia zip codes map: http://www.mapsofworld.com/usa/zipcodes/georgia/)

These typos need to be fixed to ensure accuracy of the dataset.

Issues with values in “address.street” element

By running the dataset against Atlanta-AddressStreetAudit.py, we are able to retrieve a dictionary with detailed value of address.street element that are not in the expected dictionary (step1). With the help of this dictionary, we can further update non-standard/unexpected street type with the standard ones from mapping dictionary (step2).

By examing the dictionary generated from step 1, a number of obvious issues are found and listed below

- incomplete street name
  e.g. <tag k="addr:street" v="814"/> ("id"=358781700)
  814 is actually the house number of the location. Correct street name is 'Mimosa Blvd' (location is 814 Mimosa Blvd)

- website is used as street name
  e.g. <tag k="addr:street" v="https://www.suntrust.com"/> ("id"=1894140494")
  web site of the location is used and full address is in housenumber field. Correct street name is 'Power Ferry Rd'.

- incorrect street address abbreviation (refer to "Standard address abbreviations" in the project summary)-can be programmatically cleaned by Atlanta-AddressStreetAudit.py
  e.g. <tag k="addr:street" v="Graves Spr"/> ("id"=418432069)
  Correct street name is actually 'Graves Spur'. Incorrect address abbreviation is used here.

- inconsistent direction naming in the street name-can be programmatically cleaned by Atlanta-AddressStreetAudit.py
  e.g. <tag k="addr:street" v="Village Green Cir W"/> ("id"=35177957)
  It is found most of the direction in the street name is illustrated fully such as West, Northeast. There are a few of them that are abbreviated. The above street address should be 'Village Green Cir West' to maintain consistency.

Besides the issues above, it is found that 'Standard address abbreviations' are used extensively in the street address. But no consistency found in the usage of abbreviation. For instance, '1807 preston lake dr' is used for street name for 'id'= 2442175102, while 'CENTER DRIVE' is used for 'id'=83279398. 'Cir' is used as abbreviation for 'Circle' in 78 street names, while 'Circle' is used 31 times as part of street name. In order to maintain consistency in the street naming, all standard address in the value part of the mapping dictionary (to be updated with all desired mapping) should be used. This can be achieved by running the above mentioned program in step 2.

2. Overview of the data

After cleaning the issues with the above programs, the whole file can be imported into MongoDB for additional analysis.

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

File sizes

atlanta.osm .... 556 MB
atlanta.osm.json .... 612 MB (Please note the dataset ONLY contains info from element.tag = "node" or element.tag = "way")

Number of documents contains created.user

```
db.atl2.find({"created.user":{"$exists":1}}).count()
 2690023
```

Number of nodes

```
db.atl.find({"type":"node"}).count()
2495266
```

Number of ways

```
db.atl.find({"type":"way"}).count()
194757
```

Number of unique users

```
db.atl.distinct({"created.user"}).length
745
```

Top 2 contributing user

```
db.atl.aggregate([{"$group":{"_id":"$created.user","count":{"$sum":1}}},{"$sort":{"count":-1}},{"$limit":2}])
```

```
[{ "_id" : "Liber", "count" : 1133662},{ "_id" : "woodpeck_fixbot", "count" : 636834}
```

Biggest religion (no surprise here)

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

```
[{ "_id" : "Christian", "count" : 1327}]
```

Number of users appearing only once (having 1 post)

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

3. Additional Ideas

Contributor statistics

The contributions of users seems incredibly skewed, possibly due to automated versus manual map editing (the word “bot” appears in some usernames). Here are some user percentage statistics:

- Top user contribution percentage (“Liber”) - 1133662/2690023=42.14%
- Combined top 2 users' contribution (“Liber” and “woodpeck_fixbot”) - 65.8%
- Combined Top 10 users contribution - 2310204/2690023=85.88%
- About 50% of all users contribute only 0.05% of all posts having identified user field

Atlanta dataset shows a very imbalanced contribution from the user pool. It seems the top two users are from some organizations that dedicate themselves to add info to Atlanta OpenStreetMap. Most of the users are adding only the ones that they are interested in.

Additional data exploration using MongoDB queries

Top 5 appearing amenities

```
db.atl.aggregate([{"$match":{"amenity":{"$exists":1}}},{"$group":{"_id":"$ameni-
ty","count":{"$sum":1}}},{"$sort":{"count":-1}},{"$limit":5}])
```

```
[{"_id":"place_of_worship","count":1383},{"_id" : "parking", "count" : 1204}, {"_id" : "school", "count" : 1164},{"_id" :
"parking_space", "count" : 838},{"_id" : "restaurant", "count" : 554}]
```

% of top 3 appearing amenites of total amenities

```
db.atl.find({"amenity":{"$exists":1}}).count()
8022
```

place_of_worship : 1383/8022=17.2%

parking: 1204/8022=15%
school: 1164/8022=14.5%

Top 2 cuisine types

```
db.atl.aggregate([{"$match":{"cuisine":{"$exists":1}}},{"$group":{"_id":"$cuisine","count":{"$sum":1}}},{"$sort":{"count":-1}},{"$limit":2}])
```

```
[{ "_id" : "burger", "count" : 79},{ "_id" : "pizza", "count" : 58}
```

Top 2 appearing zipcodes

```
db.atl.aggregate([{"$match":{"address.postcode":{"$exists":1}}},{"$group":{"_id":"$address.postcode","count":{"$sum":1}}},{"$sort":{"count":-1}},{"$limit":2}])
```

```
[ { "_id" : "30132", "count" : 5645},{ "_id" : "30157", "count" : 3185}]
```

Top 2 appearing counties

```
db.atl.aggregate([{"$match":{"address.county":{"$exists":1}}},{"$group":{"_id":"$address.county","count":{"$sum":1}}},{"$sort":{"count":-1}},{"$limit":2}])
```

```
[{ "_id" : "Paulding,GA", "count" : 15952},{ "_id" : "Rockdale,GA", "count" : 7575}]
```

An in-depth look at the address, postcode and county match

Postcode and id combination (combined grouping) that appeared in Paulding county in ascending order (limit by 2)

```
db.atl.aggregate([{"$match":{"address.county":{"$exists":1},"address.county":{"$in":["Paulding, GA"]},"address.postcode":{"$exists":1}}},{"$project":{"id":1,"address.postcode":1}},{"$group":{"_id":{"postcode":"$address.postcode","_id":"$id"},"count":{"$sum":1}}},{"$sort":{"count":1}},{"$limit":2}])
```

```
[{ "_id" : {"postcode”:”30101”, "_id" : “35795017”},”count”:1},{ "_id" : {"postcode”:”30101”, "_id" : “35795019”},”count”:1}]
```

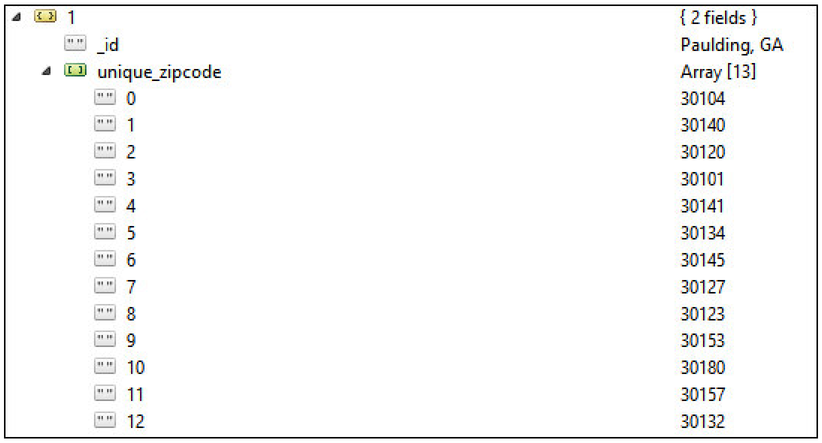
After checking the postcode 30101 and the address of the two Ids (“35795017”/“35795019”) against the Paulding county (refer to Georgia zip codes - http://www.mapsofworld.com/usa/zipcodes/georgia/), it turned out that 30101 belongs to Cobb county instead of Paulding county. So there are county errors for at least these 2 rows that need to be fixed.

These errors prompted me to further check all postcodes used in Paulding county in the dataset.

All uniques postcode appeared in rows containing Paulding county

```
db.atl.aggregate([{"$match":{"address.county":{"$exists":1},"address.postcode":{"$exists":1}}},{"$group":{"_id":"$address.county","unique_zipcode":{"$addToSet":"$address.postcode"}}}])
```

The postcodes sued in Paulding county is as follows:



```
1                         { 2 fields }
    _id                   Paulding, GA
    unique_zipcode        Array [13]
        0                 30104
        1                 30140
        2                 30120
        3                 30101
        4                 30141
        5                 30134
        6                 30145
        7                 30127
        8                 30123
        9                 30153
        10                30180
        11                30157
        12                30132
```

According to Georgia zip codes @ http://www.mapsofworld.com/usa/zipcodes/georgia/, there are only 3 postcodes used in Paulding county of Georgia. They are 30132, 30141 and 30157. So obviously the above postcodes that do not belong to these 3 are wrong and need to be audited and corrected.

From the above in-depth analysis on postcode, I propse a more in-dpeth look at the consistency across multiple fields to ensure they are aligned correctly. These should include a complete roadmap that include the following steps.
1. Identify fields that are related to each other (e.g. city, county,zip code are related entity)
2. Get the accurate database that can map these fields against each other. (e.g. Georgia zip codes @ http://www.mapsofworld.com/usa/zipcodes/georgia/)
3. Design query to fetch the results related to these fields.
4. Identify mismatch against accurate database and find out the ones need to be corrected.
5. Test the results against accurate database again to ensure all have been aligned.

Another way to improve the correctness of the postcode is to use reverse geocoding process. For the rows with coordination field (pos entity), we can search the closest address of the coordinates by using some reverse geocoding tool and then compare the detailed address info with the ones already in the dataset. In this way, we can identify mismatch in address info related to postcode, county name or city name etc. But there are several challenges to implement it. The first is the distance parameter, which is the distance from the given location which a matching address should be searched. It is one of the required inputs for a reverse geocoding tool along with the coordinate input. This input obviously depends on the density of the surrounding address. The value used in downtown area is much smaller than the ones for rural areas. The second is spatial reference used by the coordinates (usually WGS84 in decimal degrees) and most importantly how accurate the coordinates are in this spatial reference system since the accuracy of the address retrieved depends on the accuracy of the coordinate themselves.

Conclusion

After this review of the data it's obvious that there are lots of errors in the Atlanta area from OpenStreetMap. Some errors are so obvious that they can be cleaned right away, while there are many other hidden ones that need a fair amount of eff-forts to tackle them. It seems better to have correct inputs in the first place. Some automatic checking against standard accurate database such as 'Georgia zip codes' and 'Standard address abbreviations' before user can insert new info into the OpenStreetMap seems necessary to avoid future clean-up efforts. By doing this, users will be aware of the errors before updating the OpenStreetmap. But like everything else, not everything can be automatically corrected without human intervention. We still need to rely on initial auditing efforts from each and every user to ensure high-quality updates in the OpenStreetmap.