Project Data Wrangling with MongoDB

Anna Anesiadou-Hansen

2015-07-27 Mon

Contents

1 Problems encountered in the map

For the project I choose the map of Mobile in Alabama from MAPZEN metro extracts because of the city name. The city name reminds me to my company's business - mobile solutions.

To find problems in the map of Mobile I did the following steps:

1. Manual scanning of the map file was difficult because of huge amount of data.

2. Created audit_address.py to audit the street names

   In several street names were found abbreviations at the beginning or at the end of the street name. Examples:

   - Airport Blvd
   - S Mobile Ave
   - Hwy. 90 West

3. Created audit_phone.py to audit the phone numbers

   The phone numbers did not have the same format. The phone numbers notation should follow the NANP (North American Numbering Plan), see `https://en.wikipedia.org/wiki/North_American_Numbering_Plan` for more information. Examples of inconsistent/not Mobile city phone numbers:

   - 251 5108500: does not follow the NANP.
   - +2147483647: Northern Texas area code. Local area code of Mobile city is 251.)
   - +1 251 432 4117: international format

4. Created audit_postcode.py to audit the postal codes

   One incosistent postal code was found: AL 36615. Several postal codes were found that do not belong to Mobile city. Further investigations have shown that the map of Mobile covers larger areas than Mobile city. These postal codes belong to places outside Mobile city but are part of the map. Examples:

   - 36527: Spanish Fort, Alabama
   - 36526: Daphne, Alabama
   - 36532: Fairhope Alabama

1.1 Abbreviations in Street Names

To solve the problem with abbreviations at the beginning of a street name I created the function update_beginning(name) in file audit_address.py. This function uses the dictionary mapping_street_name_beginning to replace the abbreviations with their full name.

```
mapping_street_name_beginning = {
    "Hwy." : "Highway",
    "S" : "South",
    "Mc ": "Mc",
    "Dr" : "Dr.",
    "N." : "North",
    "Capt." : "Captain"
    }
```

To solve the problem with abbreviations at the end of a street name I created the function update_type(name) in file audit_address.py. This function uses the dictionary mapping_street_type to replace the abbreviations with their full name.

```
mapping_street_type = {
    "Ave" : "Avenue",
    "Blvd" : "Boulevard",
    "N" : "North",
    "S" : "South",
    "W" : "West",
    "E" : "East",
    "Dr" : "Drive"
    }
```

1.2 Phone Numbers

The format of national and international phone numbers of Mobile is reformatted according to NANP standard with the function update_phone(phone) in file audit_phone.py. For example the phone numbr 251 5108500 is reformatted to 251-510-8500.

1.3 Postal Codes

Postal codes should only contain digits. I created the function update_postcode(postcode) in file audit_postcode.py in order to remove all non-digits.

2 Overview of the Data

This section contains basic statistics about the dataset.

2.1 File sizes

- mobile_alabama.osm: 65 MB

- mobile_alabama.osm.json: 91 MB

2.2 Number of documents

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

307753

2.3 Number of nodes

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

272214

2.4 Number of ways

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

35537
```

2.5 Number of unique users

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

168
```

2.6 Top 3 contributing user

```
> db.mobile_AL.aggregate([
{ "$group" : { "_id" : "$created.user", "count" : { "$sum" : 1 }}},
{ "$sort" : { "count": -1 }},  { "$limit" : 3 }
])

{ "_id" : "woodpeck_fixbot", "count" : 178216 }
{ "_id" : "jfoote", "count" : 42129 }
{ "_id" : "ELadner", "count" : 21293 }
```

2.7 Number of users appearing only once (having 1 post)

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

{ "_id" : 1, "num_users" : 24 }
```

2.8 Top 10 amenities

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

{ "_id" : "place_of_worship", "count" : 569 }
{ "_id" : "parking", "count" : 191 }
{ "_id" : "school", "count" : 149 }
{ "_id" : "restaurant", "count" : 64 }
```

```
{ "_id" : "grave_yard", "count" : 55 }
{ "_id" : "fast_food", "count" : 55 }
{ "_id" : "bank", "count" : 31 }
{ "_id" : "fuel", "count" : 30 }
{ "_id" : "fire_station", "count" : 19 }
{ "_id" : "hospital", "count" : 16 }
```

3 Other ideas about the datasets

3.1 Contributing statistics

The contribution of users seems very skewed, possibly due to automated versus manual map editing. The word "bot" in some user names is also a hint for that. The following query lists the user contributions:

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

Here are some basic user percentage statistics:

- Top user contribution percentage ("woodpeck_fixbot"): 65%

- Combined top 3 contributors ("woodpeck_fixbot", "jfoote" and "ELadner"): 88%

- Combined top 5 users contribution: 97%

3.2 Local shops on OpenStreetMap

OpenStreetMap is a media that might not be used well by local shops in Mobile, Alabama. To verify this statement the following query is done. It lists the shop categories and the amount of shops of this category being put on OpenSteetMap.

```
> db.mobile_AL.aggregate([
{"$match":{"shop":{"$exists":1}}},
{"$group":{"_id":"$shop", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":30}
])

{ "_id" : "supermarket", "count" : 22 }
{ "_id" : "doityourself", "count" : 9 }
{ "_id" : "car_repair", "count" : 9 }
{ "_id" : "department_store", "count" : 8 }
{ "_id" : "convenience", "count" : 7 }
{ "_id" : "hairdresser", "count" : 4 }
```

```
{ "_id" : "jewelry", "count" : 3 }
...
{ "_id" : "hardware", "count" : 1 }
{ "_id" : "shoes", "count" : 1 }
{ "_id" : "pawnbroker", "count" : 1 }
{ "_id" : "medical_supply", "count" : 1 }
{ "_id" : "pet", "count" : 1 }
{ "_id" : "mall", "count" : 1 }
{ "_id" : "bowling alley", "count" : 1 }
{ "_id" : "clothes", "count" : 1 }
{ "_id" : "books", "count" : 1 }
{ "_id" : "electronics", "count" : 1 }
{ "_id" : "music", "count" : 1 }
```

The results above are very low for a city of roughly 200000 inhabitants. For example, the number of shops with clothes, shoes or electronics are only 3 shops altogether. This means that few shop owners put their shop information on OpenStreetMap. This supports the original statement. To increase substantially the number of shops included in the map a collaboration with the local university could help. Computer Information Systems of University of Mobile could have as part of their studies an obligatory project to help local shop owners to put their shop information on OpenStreetMap.

3.3 GNIS Information

The Geographic Names Information System (GNIS) is data supplied by government agencies. For a better structure of the json file mobile_alabama.osm.json all GNIS information of a node should be included in a dictionary "gnis_info" structure like it is done for address. For example gnis_info could get the shape:

```
gnis_info = { "id" : "143846", "County" : "Mobile",
              "Class" : Populated Place, "County_num" : "097",
              "ST_alpha" : "AL", "ST_num" : "01"}.
```

3.4 Conclusion

After auditing and cleaning the map data of Mobile, Alabama higher quality data for OpenStreetMap can be produced. Currently only few shops have their details on the map. A cooperation between the local university and shop owners could help to improve the situation substantially. Introducing a GNIS dictionary in all relevant nodes would improve the structure of the data model.