Open Street Map Project
Data Wrangling with MongoDB

Xinqi You

June 3, 2015

Map Area: Austin, TX, United States

https://www.openstreetmap.org/relation/113314

https://s3.amazonaws.com/metro-extracts.mapzen.com/austin_texas.osm.bz2

1. Problems Encountered in the Map

I first run the audit.py and data.py on the sample.osm of Austin Map data and found that there are several problems.

- Lack of street types ('Old 2243 West', 'Robert Martinez Jr')
- Over-abbreviated street names ('Hwy 290 W. Ste. 301')
- New dictionary 'tiger'
- Incorrect postcodes (Austin area zip codes start with 78)

Lack of Street Types

For the street names without street types, it is unreasonable to just add 'Street' or 'Road' since we do not have information about their types. Also, there are new street types such as 'Highway 290' or 'Interstate Highway 35'. My solution is just to leave them as their original names.

Over-abbreviated street names

There are a lot of street names ended with numbers. Using the regular expression search in the audit.py results in street types of numbers, locations, etc. Therefore I modify the data.py and directly substitute those abbreviated names, for example 'Hwy 290 W. Ste. 301' to 'Highway 290 West Suite 301'.

New dictionary 'tiger'

From the json output of our sample dataset, there is a new source called 'tiger' (Topologically Integrated Geographic Encoding and Referencing system). To make the json output more specific, I created a new dictionary for 'tiger' and include information of 'tiger' in it, just like the 'address'.

Incorrect Postcodes

For the postcodes that not start with 78, I drop those nodes.

2. Data Overview

This section contains basic information about the Austin dataset.

File sizes

austin_texas.osm ..... 175.6 MB

austin_texas.osm.json..... 248.5 MB

```
# Number of documents
> db.austin.find().count()
863087

# Number of nodes
> db.austin.find({'type':'node'}).count()
782010

# Number of ways
> db.austin.find({'type':'way'}).count()
81065

# Number of unique users
```

```
> db.austin.distinct('created.user').length
943
```

```
# Top 1 contributing user
> db.austin.aggregate([
...   {"$group":{"_id":"$created.user","count":{"$sum":1}}},
...   {"$sort":{"count":-1}},
...   {"$limit":1}
...   ])
{ "_id" : "woodpeck_fixbot", "count" : 238901 }
```

```
# Number of users appearing only once
> db.austin.aggregate([{"$group":{"_id":"$created.user",
"count":{"$sum":1}}}, {"$group":{"_id":"$count",
"num_users":{"$sum":1}}}, {"$sort":{"_id":1}},
{"$limit":1}])
{ "_id" : 1, "num_users" : 182 }
```

3. Additional Ideas

Contributor statistics

- Top user contribution percentage ("woodpeck_fixbot") – 27.68%
- Combined Top 2-10 users' contribution percentage – 29.85%
- Combined Top 10 users' contribution percentage – 57.53%

The contribution of users is heavily skewed towards to the automated map editing ("woodpeck_fixbot"). Top 2-10 users are clearly personal users that indicated by their user name ("varmint", "richlv", "Clorox", "Iowa Kid", "HJD", "afdreher", "Chris Lawrence", "TexasNHD", "Longhorn256"). It seems Austin users are quite avid in contributing compared with Charlotte in the sample project. Also the numbers of

contributions of the top 2-10 users are very close.

I suggest the website should display the leaderboard of contributions and show how many differences the users have. When users see their scores are so close, people tend to contribute more in order to advance others more. As a result, we would have more participation.

Nevertheless, it is highly possible that some users might submit duplicate or incorrect information to earn higher scores. We will need some cross-validation methods (such as compare with submissions from other users or the automatic contribution) to calculate their effective contributions.

Additional data exploration

```
# Top 10 Amenities in Austin
> db.austin.aggregate([
...     {"$match": {"amenity": {"$exists":1}}},
...     {"$group": {"_id": "$amenity", "count":{"$sum":1}}},
...     {"$sort":{"count":-1}},
...     {"$limit":10}
...     ])
{ "_id" : "parking", "count" : 1860 }
{ "_id" : "restaurant", "count" : 695 }
{ "_id" : "waste_basket", "count" : 591 }
{ "_id" : "school", "count" : 563 }
{ "_id" : "", "count" : 519 }
{ "_id" : "place_of_worship", "count" : 488 }
{ "_id" : "fuel", "count" : 373 }
{ "_id" : "bench", "count" : 349 }
{ "_id" : "shelter", "count" : 231 }
```

```
{ "_id" : "bank", "count" : 155 }
```

Biggest Religion

```
> db.austin.aggregate([
...     {"$match": {"amenity": {"$exists":1},
"amenity":"place_of_worship"}},
...     {"$group": {"_id":"$religion", "count":{"$sum":1}}},
...     {"$sort":{"count":-1}},
...     {"$limit":1}
...     ])
{ "_id" : "christian", "count" : 444 }
```

Most popular cuisines (Besides the null, Mexican food is the most popular)

```
> db.austin.aggregate([
...     {"$match": {"amenity": {"$exists":1},
"amenity":"restaurant"}},
...     {"$group": {"_id":"$cuisine", "count":{"$sum":1}}},
...     {"$sort":{"count":-1}},
...     {"$limit":5}
...     ])
{ "_id" : null, "count" : 372 }
{ "_id" : "mexican", "count" : 69 }
{ "_id" : "american", "count" : 28 }
{ "_id" : "pizza", "count" : 23 }
{ "_id" : "chinese", "count" : 20 }
```

Top 5 fast food

```
> db.austin.aggregate([
...     {"$match": {"amenity": {"$exists":1},
"amenity":"fast_food"}},
```

```
...     {"$group": {"_id":"$name", "count":{"$sum":1}}},
...     {"$sort":{"count":-1}},
...     {"$limit":5}
...     ])
{ "_id" : "Whataburger", "count" : 30 }
{ "_id" : "McDonald's", "count" : 30 }
{ "_id" : "Subway", "count" : 26 }
{ "_id" : "Taco Bell", "count" : 24 }
{ "_id" : "Jack in the Box", "count" : 18 }
```

```
# Most popular gas station
> db.austin.aggregate([
...     {"$match": {"amenity": {"$exists":1},
"amenity":"fuel"}},
...     {"$group": {"_id":"$name", "count":{"$sum":1}}},
...     {"$sort":{"count":-1}},
...     {"$limit":1}
...     ])
{ "_id" : "Shell", "count" : 80 }
```

4. Conclusion

From the data exploration of last section, we can see that the dataset is far from cleaned. There are a lot of either 'null' or ""(empty) values and documents that lack exact addresses in the Austin map dataset. These are the areas that require either user contributions or automatic contributions. Also notice that TIGER contribution takes up around 7.8% of the total documents.