OpenStreetMap Sample Project
Data Wrangling with MongoDB

Christopher Yñigo Velayo

Map Area: Boulder, CO

http://overpass-api.de/api/map?bbox=-105.4303,39.9097,-105.1110,40.1201
http://www.openstreetmap.org/relation/112298

1. Problems encountered in the map

Overall, this dataset, chosen because I live in this area, was a fairly clean dataset. The main problems encountered when running this data against a first version of the data.py script related to how street names were entered into the system. In addition to the problem of dealing with abbreviated street types that was solved in the Problem Set, some street names were entered without their street type (ex: Baseline instead of Baseline Road), and the list of expected street types and mappings needed to be expanded on. There was a minor problem with postal codes as seven entries had postal codes that were not in the five digit zip code format. Finally some street names began with an abbreviated direction (E. instead of East).

To solve these problems, I adapted the functions from the dataset to create a list of problematic entries, both for the street types and for the postal codes. I used this list to the update the list of expected street types for appropriate street types, the mapping dictionary which was used to update problematic entries to canonical versions, and created rules to fix the problematic postal codes and abbreviated directions.

One problem which was not solved was the issue where some users overloaded the addr:street field with the complete address, suite or apartment numbers included. As there were only a few times this problem occurred, and none occurred more than once, the judgement call was made to not attempt to programmatically fix these issues in the interest of time.

2. Overview of the data

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

File Sizes:

Boulder.osm – 76 MB
Boulder.osm.json – 85 MB

Number of entries:

```
> db.boulder.find().count()
392084
```

Number of Unique Users:

```
> db.boulder.distinct('created.user').length
527
```

Number of nodes:

```
> db.boulder.find({type:'node'}).count()
354339
```

Number of ways:

```
> db.boulder.find({type:'way'}).count()
37699
```

Number of restuarants:

```
> db.boulder.find({type:'way'}).count()
37699
```

Number of cafes:

```
> db.boulder.find({amenity:"cafe"}).count()
71
```

Number of shops:

```
> db.boulder.find({shop:{$exists:1}}).count()
403
```

Number of outdoor shops:

```
> db.boulder.find({shop:'outdoor'}).count()
6
```

3. Other Ideas about the dataset

Further exploration did show some problems that did not appear on the initial audit of the OSM file. First, a large number of zip codes are not actually associated with Boulder. Boulder zip codes begin with 803. According to MongoDB, there are 4138 entries with associated zip codes.

```
db.boulder.find({'address.postcode':{$exists:1}}).count()
4138
```

Looking at a breakdown of zip codes shows about half of the zip codes do not belong in Boulder:

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

```
{ "_id" : "80306", "count" : 1 }

{ "_id" : "80309", "count" : 2 }

{ "_id" : "80020", "count" : 8 }

{ "_id" : "80203", "count" : 13 }

{ "_id" : "80304", "count" : 14 }

{ "_id" : "80021", "count" : 30 }

{ "_id" : "80503", "count" : 36 }
```

{ "_id" : "80302", "count" : 135 }

{ "_id" : "80305", "count" : 304 }

{ "_id" : "80303", "count" : 590 }

{ "_id" : "80027", "count" : 680 }

{ "_id" : "80301", "count" : 1052 }

{ "_id" : "80026", "count" : 1273 }

What seems to have happened was that the layout of Boulder did not lend itself well to a bounding box method of exporting OSM data. Most of the non-Boulder zip codes are from the surrounding cities and towns. One way to solve this would be to use the position data when cleaning the data set to include only latitudes and longitudes within the city of Boulder.

Further, while street types were relatively easy to create a map to fix errors, street names require much more effort to catch errors. While the abbreviated direction was easy to catch, the nature of street names would make misspellings more difficult to detect and fix programmatically. For casual uses, a larger number of errors can be tolerated. More critical uses would require significantly increased attention to catch these types of errors.

4. Conclusion

Similar to the sample project, this dataset appears to be reasonably cleaned, though incomplete. It would be interesting to plot the data to see what areas might be missing and therefore in need of user attention.