Data Analyst Nanodegree Project 3 OSM

August 17, 2016

1 OpenStreetMap Sampe Project

2 Data Wrangling with MongoDB

2.1 Erik Istre

Map Area: Denver, CO, United States
https://mapzen.com/data/metro-extracts/metro/denver-boulder_colorado/

```
In [4]: from pymongo import MongoClient
        import os
        client = MongoClient()
        db = client['osm']
        collection = db.denver

        def aggregate(pipeline):
            return [doc for doc in collection.aggregate(pipeline)]
```

2.2 Problems Encountered in the Map

I noticed a few problems in the data set. 1. Inaccurate state values. 2. Inconsistent recording of street names. 1. Inconsistent abbreviations. 2. Inconsistent capitalization. 3. Problems with postal codes. 1. Inconsistent recording. 2. Inaccurate/Invalid postal codes.

2.2.1 Incorrect State Values

Some of the values listed for state for a node can't possibly be correct since the map region is contained in Colorado. Some were the result of putting a zip code for the state field, or the city name of "Denver". Sometimes the state was written as "Colorado" rather than "CO" which is preferred for the OSM data.

I corrected this by recording the standard value of "CO" for each document that had a listing for state. It did not seem wise to populate every field with this value in case it isn't really always needed. Whether to do this or not would depend on the usage case.

2.2.2 Inconsistent Abbreviation of Street Types

A portion of the street names are written with abbreviations for the street type. For example, "Crown Crest Blvd., Suite 300" uses the abbreviation "Blvd." for "Boulevard" while another value uses "Blvd". Another example is using "Ter" or "Terr" for "Terrace". This can create problems when analyzing the data since we have to be concerned with multiple abbreviations standing in for the same value.

I corrected this by applying a translation for common abbreviations of street types to the street name values. This applied if the abbreviation occurred space-separated anywhere in the string for the street name.

2.2.3 Inconsistent Capitalization

Some of the street names were written with different capitalization. This reduces the human readability of the data. I changed each street name to use a camel case standard.

2.2.4 Inconsistent Recording of Postal Codes

Some postal codes were written according to different conventions. For example there was the standard 5 digit coding as in '80234', but also the 9 digit as in '80214-1825'.

I corrected this by standardizing to the 5 digit postal code format by removing the last 4 digits for those written in the 9 digit format.

2.2.5 Inaccurate/Invalid Postal Codes

A few of the postal codes were inaccurate or invalid. For example, there was one that was 4 digits, which isn't possible. And there was another which was "Golden, CO 80401" which is more information than needed in the postal code, making it invalid.

I corrected this by verifying that all recorded postal codes were valid for the state of Colorado (between 80001 and 81658. (Since the area I chose was larger than Denver, it's error-prone to restrict the range more than this.) Further, if a zip code contained any alphabetic characters, they were stripped and then the remaining digits were checked for accuracy by being in the proper range.

2.3 Data Overview

A quick look at some basic statistics on the data set and the MongoDB queries used to generate them.

```
In [48]: # File size of MongoDB
         print("Size of denver-boulder_colorado.osm...", int(os.path.getsize('denver-boulder_colorado.o
         print("Size of denver-boulder_colorado.osm.json...", int(os.path.getsize('denver-boulder_colora
```

```
Size of denver-boulder_colorado.osm... 771 MB
Size of denver-boulder_colorado.osm.json... 854 MB
```

```
In [8]: # Number of documents
        total_docs = collection.find().count()
        print(total_docs)
```

```
3862148
```

```
In [5]: # Number of nodes
        collection.find({"doc_type":"node"}).count()

Out[5]: 3471143
```

```
In [6]: # Number of ways
        collection.find({"doc_type":"way"}).count()

Out[6]: 389235
```

```
In [7]: # Number of relations
        collection.find({"doc_type":"relation"}).count()

Out[7]: 1770
```

```
In [12]: # Number of unique users
         len(collection.find().distinct('created.user'))

Out[12]: 1871
```

```
In [21]: # User contributing the most documents
         aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                    {"$sort":{"count":-1}},
                    {"$limit":3}])
```

```
Out[21]: [{'_id': 'Your Village Maps', 'count': 657896},
          {'_id': 'chachafish', 'count': 464768},
          {'_id': 'woodpeck_fixbot', 'count': 351383}]
```

```
In [19]: # Number of users contributing only one document
         aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                   {"$group":{"_id":"$count", "num_users":{"$sum":1}}},
                   {"$sort":{"_id":1}},
                   {"$limit":1}])
```

```
Out[19]: [{'_id': 1, 'num_users': 346}]
```

```
In [20]: # Average user contribution
         # Median would be more appropriate but can't find in MongoDB
         aggregate([{"$group": {"_id":"$created.user", "count":{"$sum":1}}},
                   {"$group": {"_id":"$created.user", "avgContribution":{"$avg": "$count"}}}
                  ])
```

```
Out[20]: [{'_id': None, 'avgContribution': 2064.215927311598}]
```

2.4 Additional Ideas

2.4.1 Improving Completeness

The most difficult part of the OSM data would seem to be determining whether it was complete or not. One way to improve on this (and simultaneously improve accuracy) would be to make editing OSM more easily accessible wherever you are. The obvious way to do this would be an OSM mobile app.

This app could be programmed to prompt the user specific questions about the location they were currently in. The user could quickly answer the question and put in more information if they so desired. It could also determine whether it knew anything at all about the location the user was at and then prompt the user to create an entry. This increased access and ease of use would result in many more contributions on a regular basis increasing the accuracy and completeness of the data.

However, there is considerable difficulty in developing a reliable app in the first place. That would be a significant task which would require some time and effort.

2.4.2 Additional Data Exploration

```
In [9]: # Percent of documents from "bots"
        collection.find({"created.user" : {"$regex": "bot"}}).count() / total_docs
```

```
Out[9]: 0.09133751477157272
```

```
In [15]: # Top 5 streets
         aggregate([{"$match": {"address.street":{"$exists":1}}},
                   {"$group" : {"_id":"$address.street", "count":{"$sum":1}}},
                   {"$sort":{"count":-1}},
                   {"$limit":5}])
```

```
Out[15]: [{'_id': 'Lipan Street', 'count': 447},
          {'_id': 'West 32nd Avenue', 'count': 428},
          {'_id': 'South Parker Road', 'count': 376},
          {'_id': 'Emerson Street', 'count': 323},
          {'_id': 'West 35th Avenue', 'count': 320}]
```

```
In [16]: # Top 5 postcodes
         aggregate([{"$match": {"address.postcode" : {"$exists": 1}}},
                    {"$group": {"_id":"$address.postcode", "count":{"$sum":1}}},
                    {"$sort": {"count":-1}},
                    {"$limit":5}])
```

```
Out[16]: [{'_id': 80211, 'count': 6725},
          {'_id': 80026, 'count': 3676},
          {'_id': 80205, 'count': 2592},
          {'_id': 80204, 'count': 2539},
          {'_id': 80212, 'count': 2056}]
```

```
In [17]: # Top 5 cities
         aggregate([{"$match": {"address.city": {"$exists": 1}}},
                    {"$group": {"_id":"$address.city", "count":{"$sum":1}}},
                    {"$sort": {"count": -1}},
                    {"$limit":5}])
```

```
Out[17]: [{'_id': 'Denver', 'count': 19243},
          {'_id': 'Lafayette', 'count': 3555},
          {'_id': 'Boulder', 'count': 2889},
          {'_id': 'Broomfield', 'count': 987},
          {'_id': 'Aurora', 'count': 884}]
```

```
In [22]: # Top 10 appearing amenities
         aggregate([{"$match":{"amenity":{"$exists":1}}},
                    {"$group":{"_id":"$amenity", "count":{"$sum":1}}},
                    {"$sort":{"count":-1}},
                    {"$limit":10}])
```

```
Out[22]: [{'_id': 'parking', 'count': 14618},
          {'_id': 'restaurant', 'count': 1926},
          {'_id': 'school', 'count': 1506},
          {'_id': 'fast_food', 'count': 938},
          {'_id': 'place_of_worship', 'count': 910},
          {'_id': 'bicycle_parking', 'count': 868},
          {'_id': 'bench', 'count': 670},
          {'_id': 'fuel', 'count': 667},
          {'_id': 'shelter', 'count': 505},
          {'_id': 'bank', 'count': 485}]
```

```
In [37]: # Biggest religions
         aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"place_of_worship"}},
                    {"$group":{"_id":"$religion", "count":{"$sum":1}}},
                    {"$sort":{"count":-1}},
                    {"$limit":3}])
```

```
Out[37]: [{'_id': 'christian', 'count': 815},
          {'_id': None, 'count': 63},
          {'_id': 'jewish', 'count': 12}]
```

```
In [36]: # Some of the religious places that don't identify a religion
         aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"place_of_worship", "religion":{"$exi
                    {"$project":{"_id":0, "amenity":1, "name":1}},
                    {"$limit":3}])
```

```
Out[36]: [{'amenity': 'place_of_worship', 'name': 'Mother Cabrini Shrine'},
          {'amenity': 'place_of_worship', 'name': 'Brotherhood of the White Temple'},
          {'amenity': 'place_of_worship', 'name': 'New Hope Family Christian Center'}]
```

```
In [30]: # Top 5 cuisines
         aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"restaurant", "cuisine":{"$exists":1}
                    {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
                    {"$sort":{"count":-1}},
                    {"$limit":5}])
```

```
Out[30]: [{'_id': 'american', 'count': 188},
          {'_id': 'mexican', 'count': 171},
          {'_id': 'pizza', 'count': 116},
          {'_id': 'italian', 'count': 61},
          {'_id': 'sandwich', 'count': 57}]
```

2.5 Conclusion

The OSM data set provides an interesting view of a place since it's put there by users. While the data isn't always perfectly clean, the data is provided by people familiar with the area and has is uploaded with extra care. The analysis above suggests that some of these errors in the data are easily fixed, and so having automated bots in place can really help to improve the data. This approach lets users focus on the aspects of their data entry that are more important to them.