Wrangling OpenStreetMap Data

by Frank Corrigan

Map Area: Boulder (& Denver) County, Colorado, United States

OpenStreetMap Link: https://www.openstreetmap.org/relation/1411337
MapZen OSM Download Link: https://mapzen.com/data/metro-extracts

Overview

OpenStreetMaps.com is a communitry-built, open-source resource for geographic maps around the world. It catalogues data about roads, trails, cafés, railway stations, and much more[1]. One of the primary methods for sharing the underlying data for these maps is through xml. This data type is easy for humans to read, there is high-availability of xml-parsers that allow us to work with it, and it has a good compression ratio so it's easy to transfer. However, the advantage of readability means that xml takes a lot of time to parse. Obviously, this is a challenge for analyzing the dataset -- especially with large openstreetmap (.osm) files. This analysis details the process of downloading one such file, identifying problems & 'fixing' those issues, processing the data into a MongoDB database, and analyzing the dataset for the county of Boulder, CO.

[1] https://www.openstreetmap.org/about

1. Problems Encountered in the Map

This exercise starts at openstreetmap.com with the search tool that allows you to locate the geographical boundaries for the county of Boulder, CO., switch view to export, navigate to MapZen.com and download the .osm file for this geographic area. The file we download is called 'Denver Boulder, Colorado' and we will see that there is data not only in, but around Boulder county. Since the file is 616MB, we create a sample file taking every 15th line from the original file to explore and identify potential problems in the data. Concretely, we are looking for issues with validity, accuracy, completeness, consistency and/or uniformity. The rest of this document is based on the entire file.

The main issues identified for fixing are:

1. Inconsistent, abbreviated, unknown street types (example: St. for Street)
2. Lengthy postal codes (example: 11731-9834 rather than 11731)
3. Issues with city name (examples: 'Boulder, CO' rather than 'Boulder')

Inconsistent, Abbreviated, Unknown Street Types

The audit step for street types revealed several inconsistencies that required adjustment to improve the data's uniformity. Here are several examples of the challenges discovered and the fix:

- 'William St.' rather than 'William Street'
    - Abbreviations were expanded to full street type
- Highways called 'US Highway', 'Colorado SH', or simply 'Highway'
    - Highways were standardized as 'Highway' plus the number of that highway
- Baseline rather than Baseline Road
    - These instances were researched and adjusted to the full street name and type
- A unique instance of the full address was found in this field → '12354 West Alameda Parkway Lakewood Colorado'
    - Only the street name was used in these instances
- Most tedious problem unknown type. For instance '6001'
    - Process was laborious. Identified these as exceptions for import to MongoDB, searched for these addresses in the database, and pulled out either the associated *way* id or *node* id and used open street map to identify the true street name. For example…
        - `db.osmb_805.find({ "address.street" : '6001'})`
        - http://www.openstreetmap.org/way/223312254 showed me this location is on South Newport Street. So update record...
        - `db.osmb_805.update({"address.street" : "6001"},`

```
{ $set: { "address.street" : "South Newport Street"} } )
```

- Street type '6001' now changed to 'South Newport Street'

Lengthy, Inconsistent Postal Codes

Two and a half error types were found when attempting to standardize postal codes. First, while we wanted 5-digit postal codes, some of them were full 9-digit codes such as '80214-1803'. Second, there were instances with 6-digit codes where it appeared the final digit was repeated as such… '802377'. In order to 'fix', for any postal code containing 5+ characters the first five were used.

The last half error type was in the fact that some of these zip codes are not actually in Boulder County[2]. There are 70 zip codes that occur outside of Boulder County. This doesn't necessarily need to be corrected for the purpose of this analysis since the data is indeed for Denver - Boulder. However, it is good to know there there are many zip codes from Denver, Broomfield, and Jefferson Counties which are adjacent to Boulder County. This 'error' is seen again in city name issues.

City Name Issues

The most prevalent issue here was that cities were stored as a city, state combo. For instance, 'Boulder, CO' rather than simply 'Boulder'. In order to correct, characters after a comma were discarded. Beyond that, there were a handful of misspellings (i.e. Demver).

Looking at a list of city names, it becomes clear that some of these cities are not in Boulder County. From Wikipedia, if we consider all cities, towns, census-designated communities, and unincorporated communities in Boulder County[3], we still have several cities not within county borders including the city of Denver. In fact, a MongoDB query reveals that Denver is the top occuring city in this dataset.

```
result = db.osmb_807.aggregate([{ "$match" : { "address.city" : { "$exists" : 1 } } },
                                { "$group" : { "_id" : "$address.city",
                                               "count" : { "$sum" : 1 } } },
                                { "$sort" : { "count" : -1 } } ] )
```

And the top 5 cities are...

```
{u'count': 12869, u'_id': u'denver'}
{u'count': 3634, u'_id': u'lafayette'}
{u'count': 2588, u'_id': u'boulder'}
{u'count': 636, u'_id': u'louisville'}
{u'count': 489, u'_id': u'westminster'}
```

[2] http://www.zillow.com/browse/homes/co/boulder-county/
[3] https://en.wikipedia.org/wiki/Boulder_County,_Colorado

Denver is in Denver County and Westminster is in both Adams and Jefferson County. This leads us to believe that the map should be more aptly named 'Greater Denver - Boulder Colorado Area' rather than Denver Boulder, Colorado.

2. Data Overview

For nature enthusiasts, Boulder, Colorado is a meca. I chose Boulder County because I'd like to spend more time there -- hiking & running -- in the near future and thought it would be a good way to familiarize myself with the city before I get there. Using the mac terminal, MongoDB Python Driver and the MongoDB Shell, this section describes the dataset and some high level statistics.

File Sizes (using mac terminal)

```
ls -lh denver-boulder_colorado.osm
```

This file is 616 MB.

```
ls -lh denver-boulder_colorado.osm.json
```

This file is 671 MB.

Number of Documents (using mongo shell)

```
db.osmb_807.find().count()
```

3,226,843 documents in the database.

Number of Nodes & Ways (using pymongo driver)

```
result = db.osmb_802.aggregate([{ "$group" : { "_id" : "$type",
                                               "count" : { "$sum" : 1 } } },
                                { "$sort" : { "count" : -1 } } ] )
```

```
{u'count': 2909855, u'_id': u'node'}
{u'count': 316988, u'_id': u'way'}
```

There are 2,909,855 node elements and 316,988 way elements.

Top Contributing User (using pymongo driver)

```
result = db.osmb_807.aggregate([ { "$group" : { "_id" : "$created.user",
                                                "count" : { "$sum" : 1 } } },
                                 { "$sort" : { "count" : -1 } },
                                 { "$limit" : 1 } ])
```

```
{u'count': 611474, u'_id': u'Your Village Maps'}
```

The top contributor is 'Your VIllage Maps' with 611,474 contributions.

Number of Unique Contributors (using mongo shell & pymongo driver)

```
db.osmb_802.distinct("created.user").length
1579
```

There are 1,579 unique contributors in the database.

```
result = db.osmb_807.aggregate( [ { "$group" : { "_id" : "$created.user",
                                                 "count" : { "$sum" : 1 } } },
                                  { "$group" : { "_id" : "$count",
                                                 "num_users" : { "$sum" : 1 } } },
                                  { "$sort" : { "_id" : 1 } },
                                  { "$limit" : 1 } ] )
```

```
{u'num_users': 282, u'_id': 1}
```

Of these 1,579 unique users, 282 (18%) made only a single contribution.

3. Reflection and Additional Ideas

When I visit a new place, the first thing I want to know is where the running trails are. I would love to see a great app that uses OpenStreetMap data in order to tell you where the running trails around you are as well as A) is that a safe neighborhood to run in and B) what is the relative elevation gain. For starters, we want to know if the trailways are identified in the data. Luckily, contributors have tagged trails with either highway = footway or highway = bridleway (horse paths are also good for running) or foot = yes. Let's look at how many footways there are...

```
result = db.osmb_807.aggregate( [ { "$match" : { "highway" : "footway" } },
                                  { "$group" : { "_id" : "footway",
                                                 "count" : { "$sum" : 1 } } } ] )
```

```
{u'count': 22357, u'_id': u'footway'}
```

Awesome. The result tells us that there are 22,357 elements tagged as a footway. But, so what? What if a trail is 5 feet? This certainly starts us thinking about how the ways data is structured and what it means and how it can be used. Each ways element in the data has a list of nodes and each node has a geographic coordinate associated with it.

Let's use one of these footway ways as an example:

```
{u'node_refs':  [u'50904466',  u'50904470',  u'50904473',  u'50904474',  u'50904475',
u'50904477',   u'50904479',   u'50904482',   u'50904484',   u'50904486',   u'50904489',
u'50904491',   u'50904492',   u'50904495',   u'50904497',   u'50904499',   u'50904500',
u'50893704',   u'50904504',   u'50904506',   u'50904508',   u'50904510',   u'50904513',
u'50904514',   u'50904517',   u'50904519',   u'50904522',   u'50904524',   u'50904526',
```

```
u'50904527',  u'50904530',  u'50904531',  u'50904532',  u'50904536',  u'50904537',
u'50904539',  u'50904541',  u'50904544',  u'50904546',  u'50904548',  u'50904550',
u'50904552',  u'50904555',  u'50904557',  u'50904560',  u'50904561',  u'50904563',
u'50904564',  u'50904567',  u'50904570',  u'50904571',  u'50904572',  u'50904574',
u'50904576',  u'50904580',  u'50904581',  u'50893621',  u'50904584',  u'50904586',
u'50904588',  u'50904591',  u'50904593',  u'50904595',  u'50904597',  u'50904598',
u'50904600',  u'50904602',  u'50904643',  u'50904646',  u'50904648',  u'50904651',
u'50904653',  u'50904654',  u'50904657',  u'50904659',  u'50904661',  u'50904663',
u'50904665',  u'50904667',  u'50904669',  u'50904671',  u'50904674',  u'50904677',
u'50904679',  u'50904682',  u'50904683',  u'50904685',  u'50904687',  u'50904689',
u'50904691',  u'50904692',  u'50904695',  u'50904697',  u'50904699',  u'50904700',
u'50904703',  u'50904705',  u'50904707',  u'50904709',  u'50904710',  u'50904713',
u'50904715',  u'50904717',  u'50904720',  u'50904724',  u'50904726',  u'50904728',
u'50904730',  u'50904734',  u'50904737',  u'50904740',  u'50904741',  u'50904745',
u'50893555'], u'created': {u'changeset': u'7587522', u'version': u'2', u'user':
u'JoeGrim', u'timestamp': u'2011-03-17T17:22:57Z', u'uid': u'313700'}, u'foot': u'yes',
u'_id': ObjectId('55c091b0a5694637e228f623'), u'type': u'way', u'id': u'6147645',
u'highway': u'footway'}
```

Take the way id and find it on openstreetmap using this link:
http://www.openstreetmap.org/way/6147645

If you click on that link you can see that that way is a trail in Rocky Mountain National Park. Further, if we look at a point in that list of node refs.. we can see that it is a point on the trail:
http://www.openstreetmap.org/node/50904466

Out of scope for this project would be calculating the distance of that trail. So where does a trail like that lead? To the top of a mountain of course. We can use the 'natural' tag to search for 'peak's. We can also sort the list of peaks to reveal the tallest peak in the dataset.

```
result = db.osmb_807.aggregate( [ { "$match" : { "natural" : "peak" } },
                                  { "$project" : { "_id" : 0,
                                                   "Peak" : "$name",
                                                   "Elevation" : "$ele"} },
                                  { "$sort" : { "ele" : -1 } } ] )
```

And the tallest peak in our dataset is...

```
{u'Elevation': u'4340', u'Peak': u'Longs Peak'}
```

4,340 meters is approximately 14,240 feet. That's getting up there. There are 274 peaks[4] in Boulder

[4] http://www.climb.mountains.com/Project_Island_files/CO_County_Boulder.shtml

County according to climb.mountains.com, but our dataset only contains

```
db.osmb_807.find({"natural" : "peak"}).count()
```
387

This is yet another indication that this map spans much more than the borders of Boulder County. Another problem identified using this query was that not all peaks have associated elevation. This could be improved rather easily by looking up the elevation of those peaks in other data sources. No doubt that this information could be used by mountain rescue teams.

4. Conclusion

In order to analyze map data for the greater Denver - Boulder area, we downloaded xml data, parsed and cleaned it, and loaded it into a MongoDB database. Queries were run to determine database characteristics (# of docs, # of nodes, # of ways), document characteristics (# of users, top contributor, distribution of contributor entries), and high level statistics (# of footways, # of peaks). Data quality was the key focus during each stage of this process. The raw data is… raw. An initial cleaning process helped, but there is a ways to go to make this a valid, accurate, complete, consistent, and uniform dataset.