Santa Cruz County OpenStreetMap Data

October 17, 2016

1 Introduction

For this project, we will be auditing and cleaning OpenStreetMap data for Santa Cruz County in California. During the course, we looked at OSM data for Chicago and found inconsistencies in street name abbreviations and postal codes. We will look for the same problems in the Santa Cruz data. After cleaning it up, we will rewrite the data as a JSON file and upload it to MongoDB. We can use MongoDB to write queries on the data. A view of the area can be seen at http://www.openstreetmap.org/relation/396473.

2 Auditing the Data

2.1 Phone Numbers

Before starting with the problems from the case study, let's look at the data. If we run *auditing/auditKeys.py* we see that the node and way elements have 897 unique tag attribute keys. Cleaning all of these is beyond the scope of this project, but let's set out to clean the *phone* attributes.

When running *auditing/auditPhone.py* we see that phone numbers are not written in a standard format. Some of the numbers have +1, or parentheses around the area code, or dashes throughout the number. We want all of the phone numbers to be in the 10-digit format *xxx-xxx-xxxx*. They will be updated using *processing/updatePhone.py* when we process the file into json format.

2.2 Street Name Abbreviations

We don't want any abbreviated street names in our data. We expect the last word of each street name to be Street, Avenue, Road, etc. We'll parse through the street names in this set and create a dictionary of unexpected and abbreviated values using *auditing/auditStreetNames.py*.

We have a lot of unexpected endings, 105 to be exact. Although not all of these are invalid. Some are correct endings that we just didn't initally think of, like Circle, Highway, and Terrace. There are a few Avenues, Roads, Drives, and Streets written as Ave, Rd, Dr, and St respectively. To make the data consistent, we want to chage Ave to Avenue, Rd to Road, etc. We also want to look up Cedar, Chestnut, Front, Merrill, Pacific, and Seabright on another map and see if they are Streets, Avenues, or Roads and change them accordingly.

2.3 Postal Code Inconsistencies

We now want to look at our set of postal codes. We can write a code similar to our street name audit to print the set of postal codes in our data. There is no need for a dictionary since we want to look at the full postal code entry. Instead we'll just create the set of all postal codes by running *auditing/auditPostcodes.py*.

First thing we see is that the postcode field sometimes has full addresses, some of which are missing the actual postcode. Other entries contain the 9-digit postcode or the 5-digit postcode. Some are in the form *CA xxxxx*. Since the 5-digit form is the least common denominator, we will use regular expressions to find a 5-digit string that begins with a 9. If there is no string of this sort, the entry will be assigned *None* for its postcode. The code can be found in *processing/updatePostcode.py*.

2.4 Converting OSM to JSON

Before we upload the data to MongoDB, we have to rewrite it in JSON form. We only want to process top-level tags "node" and "way", turning their attributes in key/value pairs, except: - attributes in the CREATED array should be added under a key "created" - attributes for latitude and longitude should be added to a "pos" array, for use in geospacial indexing. Make sure the values inside "pos" array are floats and not strings. - if second level tag "k" value contains problematic characters, it should be ignored - if second level tag "k" value starts with "addr:", it should be added to a dictionary "address" - if second level tag "k" value does not start with "addr:", but contains ":", you can process it same as any other tag. - if there is a second ":" that separates the type/direction of a street, the tag should be ignored. - for "way" specifically, tags starting with "nd ref" should be added to array called "node_refs".

We will also extend street name abbreviations and add Avenue or Street to the proper street names. Postal codes, aside from 1982, will be converted to a 5-digit form. We do this by running *auditing/processMap.py*.

2.4.1 A Note on id's

When we upload our data into MongoDB, each entry will be assigned a unique id under '_id'. Our data already has an id field, but in order to use it we have to make sure that each 'id' is truely unique in our OSM file. If so, we'll have to match the id value to the '_id' key instead of a 'id' key. If we try to write code to check for uniqueness, we'll essentially have to compare over 250,000 lines against each other. This would take a ton of time to compute so instead I looked up how id's in OSM are created and found this: http://gis.stackexchange.com/questions/103572/are-osm-ids-unique-over-all-object-types. One user claims that OSM id's are composed of type of object, id, version, and combination of tags. Though it is highly unlikely, there is still a possibility that a node and way have the same id number. The larger the file, the more likely it is that we run into a problem. We'll let MongoDB assign unique numbers to the '_id' field.

This completes auditing and cleaning the data for now. We can now upload the data to MongoDB and explore the data. If necessary, more cleaning can be done there.

3 Exploring the Data

We have uploaded the data as a collection santacruz into a database called maps. There are two options for looking at the data, either through the command line or with python using the pymongo library. Since we are working in an iPython Notebook we will use pymongo. The commands are very similar if not the same.

3.1 Size

```
In [1]: from pymongo import MongoClient

        client = MongoClient()
        db = client['maps']
        db.command("dbstats")['dataSize']
```

```
Out[1]: 618670592.0
```

The size of the santacruz collection is just under 619 MB.

3.2 Entry Types

```
In [2]: import pprint

        docs = db.santacruz.find().count()
        nodes = db.santacruz.find({'type':'node'}).count()
        ways = db.santacruz.find({'type':'way'}).count()

        print 'Number of documents:', docs
```

```
        print '\nNumber of nodes:', nodes
        print '\nNumber of ways:', ways
```

```
Number of documents: 2141872

Number of nodes: 1940560

Number of ways: 201172
```

3.3 Users

```
In [3]: users = db.santacruz.distinct('created.user')
```

```
        top_users = db.santacruz.aggregate([
                {'$group': {'_id':'$created.user', 'contributions':{'$sum':1}}},
                {'$sort':{'contributions':-1}},
                {'$limit':10}
            ])

        print 'Unique users:', len(users)
        print '\nTop 10 contributing users'
        for doc in top_users:
            print doc
```

```
Unique users: 1310

Top 10 contributing users
{u'_id': u'nmixter', u'contributions': 687389}
{u'_id': u'stevea', u'contributions': 300529}
{u'_id': u'mk408', u'contributions': 208472}
{u'_id': u'DanHomerick', u'contributions': 71515}
{u'_id': u'Bike Mapper', u'contributions': 66034}
{u'_id': u'woodpeck_fixbot', u'contributions': 55896}
{u'_id': u'doug_sfba', u'contributions': 50953}
{u'_id': u'karitotp', u'contributions': 46419}
{u'_id': u'Alexander Avtanski', u'contributions': 42073}
{u'_id': u'dannykath', u'contributions': 40964}
```

3.4 Addresses

```
In [4]: postcodes = db.santacruz.aggregate([
                {'$match':{'address.postcode':{'$exists':1}}},
                {'$group':{'_id':'$address.postcode', 'count':{'$sum':1}}},
                {'$sort':{'count':-1}},
                {'$limit':15}
            ])
```

```
        print "Postal Codes"
        for doc in postcodes:
            print doc
```

```
Postal Codes
{u'count': 15874, u'_id': u'95014'}
{u'count': 753, u'_id': u'95037'}
{u'count': 470, u'_id': u'95045'}
```

```
{u'count': 425, u'_id': u'95064'}
{u'count': 335, u'_id': u'95004'}
{u'count': 236, u'_id': u'95070'}
{u'count': 100, u'_id': u'95060'}
{u'count': 92, u'_id': u'95125'}
{u'count': 87, u'_id': u'95129'}
{u'count': 74, u'_id': u'95135'}
{u'count': 73, u'_id': u'95128'}
{u'count': 59, u'_id': None}
{u'count': 54, u'_id': u'95123'}
{u'count': 49, u'_id': u'95008'}
{u'count': 47, u'_id': u'95126'}
```

```
In [5]: streetnames = db.santacruz.aggregate([
                {'$match':{'address.street':{'$exists':1}}},
                {'$group':{'_id':'$address.street', 'count':{'$sum':1}}},
                {'$sort':{'count':-1}},
                {'$limit':10}
            ])

        print "Street names"
        for doc in streetnames:
            print doc
```

```
Street names
{u'count': 348, u'_id': u'Stevens Creek Boulevard'}
{u'count': 176, u'_id': u'McClellan Road'}
{u'count': 147, u'_id': u'Rainbow Drive'}
{u'count': 130, u'_id': u'South Stelling Road'}
{u'count': 125, u'_id': u'Imperial Avenue'}
{u'count': 123, u'_id': u'East Estates Drive'}
{u'count': 120, u'_id': u'Johnson Avenue'}
{u'count': 117, u'_id': u'Miller Avenue'}
{u'count': 117, u'_id': u'Bollinger Road'}
{u'count': 113, u'_id': u'Carr Avenue'}
```

3.5 Amenities

```
In [6]: amenity = db.santacruz.aggregate([
                {'$match':{'amenity':{'$exists':1}}},
                {'$group':{'_id':'$amenity', 'count':{'$sum':1}}},
                {'$sort':{'count':-1}},
                {'$limit':10}
            ])

        print 'Top 10 Amenities'
        for doc in amenity:
            print doc
```

```
Top 10 Amenities
{u'count': 3036, u'_id': u'parking'}
{u'count': 928, u'_id': u'restaurant'}
{u'count': 617, u'_id': u'school'}
{u'count': 532, u'_id': u'toilets'}
{u'count': 521, u'_id': u'place_of_worship'}
```

```
{u'count': 449, u'_id': u'bench'}
{u'count': 438, u'_id': u'fast_food'}
{u'count': 369, u'_id': u'bicycle_parking'}
{u'count': 292, u'_id': u'cafe'}
{u'count': 243, u'_id': u'fuel'}
```

3.6 Other

```
In [7]: cuisine = db.santacruz.aggregate([
                {'$match':{'cuisine':{'$exists':1}}},
                {'$group':{'_id':'$cuisine', 'count':{'$sum':1}}},
                {'$sort':{'count':-1}},
                {'$limit':5}
            ])

        religion = db.santacruz.aggregate([
                {'$match':{'religion':{'$exists':1}}},
                {'$group':{'_id':'$religion', 'count':{'$sum':1}}},
                {'$sort':{'count':-1}},
                {'$limit':5}
            ])

        denomination = db.santacruz.aggregate([
                {'$match':{'religion':'christian','denomination':{'$exists':1}}},
                {'$group':{'_id':'$denomination', 'count':{'$sum':1}}},
                {'$sort':{'count':-1}},
                {'$limit':5}
            ])

        print 'Most popular cuisine'
        for doc in cuisine:
            print doc

        print '\nReligions'
        for doc in religion:
            print doc

        print '\nChristian denominations'
        for doc in denomination:
            print doc
```

```
Most popular cuisine
{u'count': 124, u'_id': u'mexican'}
{u'count': 88, u'_id': u'pizza'}
{u'count': 87, u'_id': u'vietnamese'}
{u'count': 86, u'_id': u'burger'}
{u'count': 79, u'_id': u'chinese'}

Religions
{u'count': 430, u'_id': u'christian'}
{u'count': 12, u'_id': u'buddhist'}
{u'count': 8, u'_id': u'jewish'}
{u'count': 2, u'_id': u'unitarian_universalist'}
{u'count': 1, u'_id': u'pagan'}
```

```
Christian denominations
{u'count': 42, u'_id': u'baptist'}
{u'count': 33, u'_id': u'catholic'}
{u'count': 23, u'_id': u'presbyterian'}
{u'count': 21, u'_id': u'lutheran'}
{u'count': 21, u'_id': u'methodist'}
```

4 Conclusion

After seeing the problems that we've encountered and due to the fact that there are over 1300 users contributing to the data, I imagine there is a lot more to audit. Running *auditing/auditAmenities*, we see that the amenities could be cleaned up by grouping similar types. For example, bar and pub represent the same type, as does coffee and cafe. We would need to think of a systematic way of fixing this, though I think the only choice would be to go in and manually group similar type.

Adding postal codes to addresses can be quite challenging. My first thought was to use street names to add postal codes, but some streets span multiple cities or multiple postal codes within a single city. Alternatively, we could look up a US postal code database and try to write code to match some entries.

We will always need to audit and clean our data, but the process can be made easier if we have accurate user input. I suggest that anyone interested in inputing or cleaning OSM data for Santa Cruz to get in touch with users in the area. It sounds like a fun project to take on as a hobby!