Project 3: Data Wrangle OpenStreetMap Data

Christine Stoller

November 11, 2015

For this project, I downloaded the Metro Extract for Cincinnati, Ohio, USA, from MapZen on November 5, 2015. This OSM XML file, referred to in this project by the filename 'cincinnati_ohio.osm', contains data from the OpenStreetMap project.

As such, this data is subject to error due to human entry and the importing of data from a variety of sources, some of which may not be reliable. I identified some of the problems with this dataset, created and implemented a plan for cleaning the data, and imported it to MongoDB as a JSON file for checking.

1 Problems Encountered in the Map

The four problems I discovered in my audit of the dataset were with street types, state labels, postal codes, and city names. I addressed these problems by cleaning the XML data programmatically before converting it to JSON. This was done by running the original cincinnati_ohio.osm file against clean_cinci.py.

Street types

After modifying the code developed in Lesson 6 of *Data Wrangling with MongoDB*, I found inconsistent abbreviations and capitalizations of street types, as well as a few typos. I replaced abbreviations with their complete street types (e.g., Rd and Rd. were changed to Road) and fixed typos.

State labels

My audit of the state labels revealed more typos and inconsistent abbreviations. In this case, I converted all state labels to the appropriate two-letter abbreviations for Ohio, Kentucky, and Indiana.

Postal codes

In the interest of consistency and ease of running MongoDB queries, I removed the four-digit suffix where applicable. Most applications outside of delivering mail do not require the use of this extra identifier.

City names

I discovered more typos in the field for city, as well as misuses of the field. Namely, some entries included state information in the city field; for such entries, I moved the state information from the city field to a new state field. Furthermore, I corrected the typos and other entries whose city field listed both Cincinnati and the name of a suburb by determining which city name was applicable.

2 Overview of the Data

The size of the original XML file used in this project is approximately 412 MB, and the size of the resulting JSON file is about 476 MB. The following are additional statistics about the dataset, obtained by querying the MongoDB database using PyMongo.

```
from pymongo import MongoClient
client = MongoClient('localhost:27017')
db = client.course

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

print "Number of documents: ", docs
print "Number of nodes: ", nodes
print "Number of  ways: ", ways
```

Number of documents: 2255747
Number of nodes: 1985219
Number of ways: 270488

These totals are reasonable based on the size of the datafile.

```
user_list = db.cinci.distinct("created.user")
print "Number of distinct users:", len(user_list)
```

Number of distinct users: 565

We can compare this value to the number of distinct users found in ‘node’ and ‘way’ elements in the XML file using a modification of code from the course prior to cleaning. As we will see, the two values agree, increasing our confidence in the cleaning and import process.

```
import xml.etree.cElementTree as ET
from collections import defaultdict

def get_user(element):
    return element.attrib["uid"]

def process_map(filename):
    users = defaultdict(int)
    for _, element in ET.iterparse(filename):
        if element.tag == "node" or element.tag == "way":
            user = get_user(element)
            users[user] += 1
    return users

users = process_map('cincinnati_ohio.osm')
user_count = len(users)
print "Number of distinct users: ", user_count
```

Number of distinct users: 565

3 Other Ideas about the Dataset

Additional queries

Let’s examine some more interesting queries. In my initial audit, I found three types of keys that were very common in both ‘node’ and ‘way’ elements: highway, service, and amenity.

The highway key provides categorical information about roads and pathways. With regard to highway elements, the service key gives more detail about those marked as highway=service, meaning that they a road used to access a building, parking area, business park, and the like. The amenity key classifies community facilities, like schools, restaurants, government buildings, and so on.

I ran the following queries using PyMongo; selected results are shown below each query.

```
pipeline = [
    { "$match" : { "highway" : { "$exists" : 1 } } },
    { "$group" : { "_id" : "$highway",
                   "count" : { "$sum" : 1 } } },
    { "$sort" : { "count" : -1 } },
    { "$limit" : 15 } ]
highway= db.cinci.aggregate(pipeline)
```

```
{u'_id': u'residential', u'count': 29517}
{u'_id': u'service', u'count': 25200}
...
{u'_id': u'bus_stop', u'count': 3158}
...
{u'_id': u'motorway', u'count': 898}
{u'_id': u'primary', u'count': 622}
...
```

```
pipeline = [
    { "$match" : { "service" : { "$exists" : 1 } } },
    { "$group" : { "_id" : "$service",
                   "count" : { "$sum" : 1 } } },
    { "$sort" : { "count" : -1 } } ]
services = db.cinci.aggregate(pipeline)
```

```
{u'_id': u'driveway', u'count': 5249}
{u'_id': u'parking_aisle', u'count': 3313}
...
```

```
pipeline = [
    { "$match" : { "amenity" : { "$exists" : 1 } } },
    { "$group" : { "_id" : "$amenity",
                   "count" : { "$sum" : 1 } } },
    { "$sort" : { "count" : -1 } },
    { "$limit" : 20 } ]
amenities = db.cinci.aggregate(pipeline)
```

```
{u'_id': u'parking', u'count': 2346}
...
```

One observation we can make from these results is that the Cincinnati / Northern Kentucky region is quite automobile-friendly - even the public transportation system consists almost entirely of buses that use the same roads that cars do. Like many cities in the U.S., Cincinnati's history includes a lot of movement away from the city center and into growing suburbs. This has resulted in the need for personal automobiles to get around, which comes with it a greater need for infrastructure to support vehicle traffic and parking.

Of course, as people move back toward the city center as has been a recent trend, the road infrastructure statistics won't go down. However, we should see increases in public transportation infrastructure reflected in these data in the years to come as Cincinnati's streetcar system gets up and running.

As those changes are made, real estate developers may find the data helpful in exploring locations for new entertainment venues and residences near the streetcar line. Public transportation officials could use the data in tandem with ridership statistics in planning expansions of the streetcar system.

Additional improvements

While working with this OpenStreetMap data, I found keys with the prefixes 'tiger:' and 'gnis:' on many records. These keys provide map information that was imported from other sources: Topologically Integrated Geographic Encoding and Referencing (TIGER) data from the US Census Bureau; and the USGS Geographic Names Information System (GNIS). The OpenStreetMap Wiki describes some of the problems with these data imports.

The TIGER data was uploaded in 2007, but the data itself was from 2005. There is a TIGER fixup page with suggestions for fixing bad TIGER data in the map. OpenStreetMap does not plan to import TIGER data again, choosing instead to trust the growing base of map contributers in the US.

The GNIS data, while rich with information about geographic features in the US and Antarctica, was imported into OSM in 2009 without accounting for any outdated features. As a result, many GNIS features in the maps actually don't exist anymore. Instructions for cleaning up GNIS data appear on the OSM Wiki page.

With these things in mind, some programmatic cleaning of these entries OSM data may be possible. Many records contain user-created address and location data as well as TIGER and/or GNIS data. One could improve the quality of this data in terms of consistency and uniformity using the tools covered in the *Data Wrangling with MongoDB* course. A cross-field audit that compares information among these three types of fields would likely reveal numerous inconsistencies. Some of these conflicts could be cleaned programmatically by applying the guidelines for fixing TIGER and GNIS data. On the other hand, some conflicts may require more research and manual fixing to reconcile, especially in the case of outdated features.

4 Conclusion

While OpenStreetMap data lends itself to many applications for analysis, we have seen that it brings with it quality issues with consistency, uniformity, and accuracy. While not discussed explicitly in this project, the Cincinnati OSM data also has problems with completeness. By the nature of OpenStreetMap, this is best rectified with increased participation from locals. Many Cincinnatians are of German descent, so perhaps we will follow the example of many German cities and start an OSM Stammtisch of our own someday!