OpenStreetMap Project
Data Wrangling with MongoDB

Perry Radau
December 13, 2015

Map Area: London, Ontario, Canada
Link: https://www.openstreetmap.org/node/485248691
Data source: https://mapzen.com/data/metro-extracts/
Downloaded on December 12, 2015
Environment: Python 3.4.3 :: Anaconda 2.2.0 (x86_64)

Overview:
data.py was used for auditing the osm file data and writing out the json file.
mapdb.py was used for reading the collection from mongodb and calculating statistics.
sample-london2.osm was the shortened version of the original London,ON region data.

Problems with the Data

1. Postal codes. Some have a space between the first 3 characters and the final 3, others do not.
In addition to correcting this, I checked that the pattern matched the one used in Canada. i.e. The format should be:
<letter><number><letter><space><number><letter><number> (e.g. N6G 5E3)
Also checked that neither extra spaces or unknown characters are in the postal code.
E.g. The postal codes 'N25 0C1' and 'N6H' were found which are not valid and therefore replaced by null.

2. Street types in the address. As in the problem 6 exercises I updated the street types to ensure that abbreviations were not used (e.g. Avenue replaced Ave.), however, most were as expected.

3. Street types in the name. Much of the street information was found to be in the 'name' of the 'way'. However, not all 'name' fields are roads so this was cross-checked by ensuring that the way also had 'lanes' field which should be a number indicating road lanes. E.g. I found 'name' fields called "7-Eleven" and "A & W" which should not be treated as street names.

Specific cleaning steps:

- I replaced abbreviations for direction (e.g. North replaced N.)
- I removed extra spaces between the street name and the type.

- I updated the street type correctly when the name had a number at the end. E.g. Rd. 100 becomes Road 100.
- I updated the street type correctly when the name had a direction at the end. E.g. Wonderland Rd. N becomes Wonderland Road North.

Import to Mongodb

To import the saved json file output by the python auditing program I first started the server:

```
mongod --dbpath /usr/local/data/db
```

and in a separate terminal imported the json data file:

```
mongoimport --db users --file sample-london2.osm.json
mongo reported:
2015-12-13T20:44:38.895-0500  imported 282439 documents
```

Statistics

The mapdb.py was used for manipulating the collection stored in mongodb.
The output using the sample-london.osm (with highlighting for items requested in rubric):

JSON File size is: 60.04937934875488 MB
Count the nodes and ways:
Count of node: 267019
Count of way: 15420
Count of all records: 282439

Count of all unique users: 245

```
Top ten users by created content:
User CanvecImports created nodes/ways: 176072
User PurpleMustang created nodes/ways: 54174
User rw__ created nodes/ways: 27050
User Cecil Coxen (Township of Malahide) created nodes/ways:
3542
User fbax created nodes/ways: 3250
User fuego created nodes/ways: 2967
User permute created nodes/ways: 2904
User andrewpmk created nodes/ways: 2145
User geobase_peterson created nodes/ways: 1853
User John Peterson created nodes/ways: 1656

Hospitals (named)
LPH: User fbax created 51 node references.
```

Number of named hospitals: 1

```
Cafes:
```

Number of cafes: 8

```
Balzac's: User permute created the tag.
Starbucks: User PurpleMustang created the tag.
```

```
Starbucks: User marcaccioc created 5 node references.
The Red Door Cafe: User PurpleMustang created 9 node
references.
Tim Hortons: User andrewpmk created the tag.
Tim Hortons: User andrewpmk created the tag.
Tim Hortons: User andrewpmk created the tag.
Tim Hortons: User andrewpmk created 8 node references.
```

Other ideas about the datasets

1. Hospital/clinic finding. The raw OSM data could be improved to be more useful for hospital/clinic finding by adding more info, especially GPS coordinates, for each of the walk-in clinics as well as hospitals. Assuming that this data existed for the region, then the user would be able to create useful queries that would tell the user which hospital/clinic is closest to the user location, and the clinic address.
There is public information about hospital locations in Ontario, and specifically London and region here:
http://www.health.gov.on.ca/en/common/system/services/hosp/southwest.aspx
The regular address information in such databases would need to be parsed and formatted in the format required by OSM as described here:
http://wiki.openstreetmap.org/wiki/Tag:amenity%3Dhospital
Similarly, there is very little information about walk-in and other clinics available in the London region. (Only one with the proper tag in the full 500MB dataset.) There are some online resources such as
http://www.walkinhealth.ca/directory/walk-in-clinic/on/london/
which require conversion to the OSM format for the "clinic" amenity tag.
http://wiki.openstreetmap.org/wiki/Tag:amenity%3Dclinic

2. Postal codes and business research. If a user is interested in opening a cafe it would be useful to know the number of cafes in each subregion, eg. Grouped by postal code. However, in this data there are few cafes in total, and almost none have postal codes. The developer could generate postal codes for given GPS locations in order to realize this goal. However, there is still a need for more extensive input data regarding cafes.
Regarding the postal codes, there is geocoding available for Canadian postal codes here in the Shapefile format.
http://geocoder.ca/?freedata=1
According to wikipedia, " The **shapefile** format is a popular geospatial vector data format for geographic information system (GIS) software." Therefore it is feasible that the postal code data could be converted to a lookup table, with input of GPS coordinates providing output of a postal code. In this way a postal code could be assigned to any address (e.g. cafe or other business) in the OSM data. By

this means we could write a program to query for all the cafe coordinates, and then provide a geographic visualization of density of the cafes by postal code. Another use of the postal code lookup table would be to fill in the missing postal codes which may be needed for users who wish to construct mailing addresses from arbitray locations within the OSM (e.g. for mass mailing).

3. Finding nearest amenity. An interesting application of the data would be to use user input GPS coordinates, and a specified amenity (e.g. ”hospital”), and then the user could quickly locate the closest one by a search in the database for the nearest match. Ideally this would be a smartphone app and the database would reside in the phone for the local region. (A new one could downloaded when the user travels into a new region.) The algorithm would then calculate the direct (or street) distance to each matching amenity in the nearby region and display the top 5 to the user. This could be convenient (for cafes and restaurants) and life-saving for emergency services.