Open Street Map Cleaning Project

William McKee

July 2017

In this investigation, I will clean the Open Street Map for my home city, Philadelphia, Pennsylvania, U.S.A. and then query the data set to find out several properties about the city. This investigation presents an opportunity to improve the Open Street Map Data for Philadelphia.

I browsed the web and found Philadelphia's Open Street Map data at the URL below. I found the "Raw OpenStreetMap datasets" and downloaded the "OSM XML" file. I then unzipped the file on my computer.

https://mapzen.com/data/metro-extracts/metro/philadelphia_pennsylvania/

Open Street Map Data Overview

The uncompressed OSM file is very large (680 MB) so I reduced the file so that I can investigate and understand the file contents. I ran the open_street_osm_sample.py code provided by Udacity in the Project Details section to reduce the file size. This code also creates the sample OSM file required for project submission. I checked the sample file in order to get a rough idea of its contents.

I took special note of the <tag> elements which contain the key-value pairs that need to be cleaned. From the appearance, the most common keys were "addr", "amenity", and "place". However, a manual exploration of the sample OSM did not tell me enough information about the data set. I decided to run the Udacity Case Lesson process_map() code on the full OSM file, create the CSV files, and import into the sqlite3 database.

I took note of the most common keys from nodes_tags and got the following results.

```
sqlite> select key,count(*) from nodes_tags group by key order by
count(*) desc limit 20;
```

I also queried the `ways_tags` table. I examined the contents of the common tags in the sample OSM file and decided to clean the following fields:

`highway`, `amenity`, `street`, `city`, `state`, `postcode`, and `shop`

Following this analysis, I removed the CSV files and database I created earlier since this database contains the original data. Final queries are run on a database containing cleaned data.

Open Street Map Problems

I noticed several problems with the Open Street Map dataset.

(1) Street values are inconsistent and need to be standardized. There were plenty of street with types like “Street”, “Avenue”, “Road”, and “Place”. Unfortunately, there were also abbreviations such as “St”, “Ave”, “Rd”, “Pl”. Some were lowercase (such as “st”). We need all streets to be “Street” in order to properly query for the number of “Streets”.
(2) Street values also contain inconsistent directional notations such as “N” for “North” – so “N 5th Street” instead of “North 5th Street”.
(3) Street values sometimes contains extra information that belongs in other fields. “Spring Garden Street Philadelphia” should have “Spring Garden Street” as the street and “Philadelphia” as the city. I noticed route numbers and apartment buildings in the street field!
(4) Some street values have no type – so “Chestnut” instead of “Chestnut Street”.
(5) Some key-value pairs have values outside of the expected values as defined on the Open Street Map Wiki “Map Features” page. For example, there are tags with key = “amenity” and value = ”swimming_pool”. However, the Map Features page asks users to use a tag with key = ”leisure” and value = ”swimming_area” for swimming pools. There are non-compliant values for the keys highway, amenity, and shop.
(6) There are city, state, and postcode values outside of the expected formats or values for the Philadelphia, PA area. States can be standardized as “PA”, “NJ”, and “DE”. All variants (and misspellings) of Philadelphia can be standardized as “Philadelphia”. All postcodes can be standardized as five digit values.

Cleaning Streets

The street types are the most challenging to clean since the values are more complex than other fields. A standard Python dictionary can be used to store the mappings for street types (“St”: “Street”) and directional (“N”: “North”). There were no issues with this cleaning. Unusual street names, however, represent only a small portion of the street values (roughly 10%) but represent over 80% of the values that must be cleaned.

I checked for streets ending with numbers. If the value has a route indicator (“Route 73”), I left that value alone since this can be a valid street name. If there was no route indicator (“Loretto Avenue #8325”), then I interpreted this as extra data. The cleaning code looks for the last street type and removes the rest of the code. I had to account for values with two street types – such as “Street Road” located in Philadelphia’s northeastern suburbs so the last street type is found by the code.

In addition, I had to account for values without a street type. I added street types for a few known values but need to look at lots of maps to find other known values. I found that most of these types of values were for numbered streets such as “North 5th” so added the known “Street” type for these numbered streets. The source code open_street_clean_streets.py file, clean_unusual_street_type() function handles this cleaning but does not contain a complete set of the improvements that could be made.

Cleaning Other Types

For highway, amenity, and shop keys, I executed a simplified version of the audit_street_type() function so that I could see a list of unexpected values for these keys.

```
KEY_TYPE = '<key_type>'
EXPECTED = ['<List of expected values from Map_Features page in Open
Street Wiki>']

def audit_type(types, key, value):
    types = list containing values not in expected list
    key = current element tag "k" field
    value = current element tag "v" field
    # Check the key type
    if key == KEY_TYPE:
        # Add unexpected values
        if value not in EXPECTED:
            types.add(value)
```

For highways and shops, I was able to clean the most common types using a mapping. However, there were three unexpected types for amenities that I did not believe are unexpected. These are `swimming_pool`, `public_building`, and `gym`. I checked the Map Features in the Open Street Map Wiki and discovered that these facility types are supposed to be indicated with a different key-value tag! The OSM map has key-value pairs that are no longer used!

The open_street_clean_amenities.py file, clean_amenity() function updates both keys and values according to a pair of mappings. The new key-value pairs are the following:

Old Key-Value Pair	**New Key-Value Pair**
k=amenity; v=swimming_pool	k=leisure; v=swimming_area
k=amenity; v=public_building	k=office; v=government
k=amenity; v=gym	k=leisure; v=fitness_centre

For city, state, and postcode, I standardized the values. For example, I capitalized each letter of the state value and converted to the appropriate two letter abbreviation using a mapping.

Basic OSM Statistics

Following the cleaning, I wrote all the information to CSV files via the open_street_process_map.py file, shape_element() function and then imported to a sqlite3 database. I ran several queries on the imported data. Some highlights are presented here. Details are shown in the Open Street Map Queries.txt file.

The file sizes for the CSV and database files are as follows:

```
philadelphia_pennsylvania.osm 680335 KB
open_street_map.db            386672 KB
nodes.csv                     252593 KB
nodes_tags.csv                 11049 KB
ways.csv                       18766 KB
ways_nodes.csv                 85187 KB
ways_tags.csv                  49194 KB
```

The number of elements within each of the sqlite database tables are listed below.

```
sqlite> select count(*) from nodes;
3141843

sqlite> select count(*) from nodes_tags;
279563

sqlite> select count(*) from ways;
328006

sqlite> select count(*) from ways_nodes;
3842368

sqlite> select count(*) from ways_tags;
1504158
```

A query for the number of unique users reveals that there are 2284 users. User "dchiles" has the most entries (794391)! User "GreyTK" has the 20th most entries with 23169. 900 of the 2284 users has five or fewer entries in the data set. 481 users entered only one entry in the data set! I am not surprised by this distribution since some updates are automated and some are manual.

```
sqlite> select count(distinct(user)) from
   ...> (select user from nodes union all
   ...> select user from ways);
2284
```

Detailed OSM Statistics

The Philadelphia OSM contains 177572 "highway" types. The most common highway value is residential with 71999. The second most common highway type is service with just 38143.

There are 17093 "amenity" types in the data set. The top five are parking, school, restaurant, place_of_worship, and fast_food. I was somewhat surprised to see so many parking entries, but nearly every amenity type provides some parking so those may have been added when the amenity was added. I queried "parking" further (see Open Street Map Queries.txt – "Amenities" – query #3) and noticed the lack of parking entrances, exits, and spaces in the database. Users may have entered "parking" as a catch all for these types.

There are 3277 “shop” types in the data set with convenience being the most common (466) and supermarket being the second most common (272).

There are 16056 “addr:city” types in the data set with 3722 entries for Philadelphia. In a major surprise, Pemberton, NJ, a small town with population less than 28000 people, has the second highest entries with 2092. Trenton City, the capital of NJ, is third with 1611. I explored the Pemberton entries further (see Open Street Map Queries.txt – “Cities” – query #3) and seen that one user (“Luisi Mouse”) entered all of the information for Pemberton plus a few entries for adjacent Mount Holly.

Among states, there were 6383 PA entries, 4400 NJ entries, and 84 DE entries. There were four erroneous entries from other states. How many unique cities per state are in the Philadelphia data set? There are 129 of them in Pennsylvania, 73 in New Jersey, and just five in Delaware.

Detailed Streets Statistics and Possible Improvements

The hardest field to clean was for “addr:street”. Not surprisingly, this field presents the most improvement opportunities. I discovered this improvement need when querying street types. How many streets do we have? Are these values below really correct? 512 “Streets” seems reasonable, but only one “Market Street” is not correct for Philadelphia and its suburbs. I know that there is at least one more Market Street – in Marcus Hook, PA.

```
sqlite> select count(distinct(tags.value)) from
   ...> (select * from nodes_tags union all
   ...> select * from ways_tags) tags
   ...> where tags.type = 'addr' and tags.key = 'street' and
tags.value like '%Street';
512
```

```
sqlite> select count(distinct(tags.value)) from
  ...> (select * from nodes_tags union all
  ...> select * from ways_tags) tags
  ...> where tags.type = 'addr' and tags.key = 'street' and tags.value like 'Market Street';
1
```

How can we account properly for different “Market Streets”? After some trial and error, I decided to pair the city and street into one variable, then perform a self-join. Pairing city and street into one column allows “Marcus Hook/Market Street” to be a value for one row, and for “Philadelphia/Market Street” to be a value for another row. Now there are two separate Market Streets. If we run the query on the pairing, we see eight unique “Market Streets”.

```
sqlite> select count(distinct(pairing)) from
   ...> (select (c.value || '/' || s.value) as pairing from
   ...> (select * from nodes_tags union all select * from ways_tags)
as c,
   ...> (select * from nodes_tags union all select * from ways_tags)
as s
   ...> where c.type='addr' and c.key='city'
```

```
   ...> and s.type='addr' and s.key='street'
   ...> and s.value='Market Street'
   ...> and c.id=s.id);
8
```

When we pair the city and street, we can better query the number of street types. According to the queries in Open Street Map Queries.txt (see "Streets" – query #5), there are 532 "Streets", 430 "Avenues", 418 "Roads", 69 "Lanes", and just 13 "Places".

I also paired city and state in order to get the correct number of unique cities (211 – see Open Street Map Queries.txt – "Cities" – query #4)

Should the pairing be added to the CSV files and imported into the database? Certainly, the queries will be simplified for city/street and city/state pairs. However, there will be redundant information with some risk for inconsistent – and therefore – incorrect information.

Conclusion

The cleaned data from this investigation can be used to standardize Philadelphia's Open Street Map data and change some key-value pairs from an old convention to the latest convention. In addition, this investigation allows us to discover which areas are near completion (such as Pemberton, NJ) and which are missing information.