Udacity Data Analyst Nanodegree

P3: Wrangle OpenStreetMap Data

August 28, 2019
Preston Hall

Context

This notebook is my project 'Wrangle OpenStreetMap Data' submission. 'Data wrangling' is what the process of collecting and cleaning data is often called. Having completed the project means I am able to:

- Assess the quality of the data for validity, accuracy, completeness, consistency and uniformity.
- Parse and gather data from an xml file.
- Process data from a very large file that can be cleaned programmatically.
- Store, query, and aggregate data using SQL.
- OpenStreetMap (OSM) is built by a community of mappers that contribute and maintain data about roads, buildings, restaurants, railway stations and much more, all over the world. Since any person can register on OSM and start editing a map, it is only natural that some of the data is not valid, accurate, complete, consistent or uniform.

Map Area

Beaverton, OR, United States

- This map is of where I currently live. This is a fairly new area for me so I'm more interested to see what database querying reveals, and I'd like an opportunity to contribute to its improvement on OpenStreetMap.org (https://www.openstreetmap.org/search?query=Beaverton%2C%20or#map=12/45.4848/-122.8055).

Since the file size was quite large, I used the sample.py (https://github.com/Prestonhall31/Data-Wrangling/blob/master/sample.py) script to produce a small sample of the data was generated to make it easier to audit and iterate through the data.

Understanding the data

I wanted to know what type of data I would be working with. Running the script mapparser.py (https://github.com/Prestonhall31/Data-Wrangling/blob/master/mapparser.py) creates a dictionary with different elements and the number of occurences within the dataset that I will be working with.

```
{'bounds': 1,
 'member': 11834,
 'meta': 1,
 'nd': 402993,
 'node': 352195,
 'note': 1,
 'osm': 1,
 'relation': 429,
 'tag': 223951,
 'way': 43527}
```

Tags are child elements of nodes, ways and relations, and consist of two attributes, a key 'k' and a value 'v', that describe the meaning of the particular element to which they are attached. The script tags.py (https://github.com/Prestonhall31/Data-Wrangling/blob/master/tags.py) allows us to have a look at the most common tags in the sample dataset:

```
{'lower': 128020,
'lower_colon': 94441,
'other': 1490,
'problemchars': 0}
```

Problems Encountered in the Map

Auditing the data

Using regular expressions, street names' endings and beginnings were audited with the audit.py (https://github.com/Prestonhall31/Data-Wrangling/blob/master/audit.py) script to return any addresses that were not in the expected list.

```
{'Ave': {'Southwest 193rd Ave'},
 'GLN': {'Southwest Malcolm GLN'},
 'Gracie': {'SW Gracie'},
 'Hwy': {'SW Tualatin Valley Hwy'},
 'Rd': {'SW Canyon Rd'},
 'St': {'SW 5th St', 'SW Inglewood St', 'SW Parkway St'}}
```

Data Cleaning

In this step, by running the data.py (https://github.com/Prestonhall31/Data-Wrangling/blob/master/data.py) script, the problems encountered in the map were cleaned and data was converted from XML to CSV format. The data is transformed from document format to tabular format which makes it possible to write the data to .csv files.

Importing to database

Finally, using the createdb.py (https://github.com/Prestonhall31/Data-Wrangling/blob/master/createdb.py) file, the cleaned .csv files were imported into a SQL database using a given schema. The osm.db database has five tables:

- nodes
- nodes_tags
- ways
- ways_tags
- ways_nodes

Inspecting cities

I wanted to work within the city of Beaverton. So it is worth checking if there are other cities that are included in the dataset.

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'city'
GROUP BY tags.value
ORDER BY count DESC;
```

```
Aloha|8985
Beaverton|8323
Portland|4280
Hillsboro|121
Beaverton, OR|13
```

More than have of the values are not Beaverton, and 13 that are labeled incorrectly as Beaverton, OR. These will need to be removed.

```
DELETE FROM nodes_tags WHERE key = 'city' and value != 'Beaverton';
DELETE FROM ways_tags WHERE key = 'city' and value != 'Beaverton';
```

Data Overview

File sizes

```
Beaverton.osm ............... 85.3 MB
osm.db ...................... 47.1 MB
nodes.csv ................... 33.7 MB
nodes_tags.csv .............. 521 KB
ways.csv .................... 9.7 MB
ways_nodes.csv .............. 7 MB
ways_tag.csv ................ 2.9 MB
```

Number of nodes

```
SELECT COUNT(*) FROM nodes;
```

```
352195
```

Number of ways

```
SELECT COUNT(*) FROM ways;
```

```
43527
```

Number of unique users

```
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
```

```
245
```

Top 10 contributing users

```
SELECT e.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
GROUP BY e.user
ORDER BY num DESC
LIMIT 10;
```

```
Peter Dobratz_pdxbuildings|217663
baradam|86399
Peter Dobratz|22418
Grant Humphries|15438
Mele Sax-Barnett|11526
Darrell_pdxbuildings|7294
cowdog|6158
lyzidiamond_imports|4018
Paul Johnson|3225
Brett_Ham|2944
```

Number of users appearing only once (having 1 post)

```
SELECT COUNT(*)
FROM
    (SELECT e.user, COUNT(*) as num
     FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
     GROUP BY e.user
     HAVING num=1)  u;
```

```
51
```

Biggest religions

```
SELECT ways_tags.value, COUNT(*) as num
FROM ways_tags
    JOIN (SELECT DISTINCT(id) FROM ways_tags WHERE value='place_of_worship')
a
    ON ways_tags.id=a.id
WHERE ways_tags.key='religion'
GROUP BY ways_tags.value
ORDER BY num DESC;
```

```
christian|33
buddhist|2
spiritualist|1
```

Christian denominations

```
SELECT b.value, COUNT(*) as num
FROM ways_tags
    JOIN (SELECT DISTINCT(id) FROM ways_tags WHERE value='place_of_worship')
a
    ON ways_tags.id=a.id
    JOIN (SELECT DISTINCT(id), value FROM ways_tags WHERE key = 'denominatio
n') b
    ON a.id = b.id
WHERE ways_tags.key='religion' AND ways_tags.value = 'christian'
GROUP BY b.value
ORDER BY num DESC
LIMIT 10;
```

```
baptist|6
lutheran|4
catholic|3
methodist|3
free_methodist|1
greek_orthodox|1
mormon|1
presbyterian|1
scientist|1
unity|1
```

Most popular cuisines

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC;
```

```
chinese|4
italian|4
mexican|3
pizza|3
vietnamese|3
korean|2
sushi|2
thai|2
american|1
german|1
mediterranean|1
pasta|1
persian|1
```

Conclusion

This is a great dataset considering it has been edited and contributed by 245 different users for this city alone. It is extremely easy to input incorrect or mistyped information for a dataset this large. Knowing how to audit, clean, and update the data is a very useful skill when working on dataset of this magnitude. Although the dataset is in generally complete, valid, and accurate, its quality regarding consistency can be improved.