Wrangling and Exploring OpenStreetMap Data

Map Area

Delaware, US

- OpenStreetMap file (compressed) (.osm.bz2) : geofabrik.de | **14.7 MB**
- OpenStreetMap file (uncompressed) (.osm) : `delaware-latest.osm` | **171 MB**
- Test data file (subset of the full uncompressed .osm file) : `test_data.osm` | **29.9 KB**

I was initially interested in the map data for New York City but after downloading the corresponding OSM file and performing some preliminary exploration quickly realized that the I was short of computation resources as execution of any kind of function took a long time for such a large dataset.

Consequently, I went back to the Open Street Map website and downloaded the map data for some other regions and finally settled for the state of **Delaware**. The parsing, auditing and other steps can however be used for the map data of other regions with minor modifications.

Problems Encountered in the Map

Performing some basic exploration of the map data, I noticed following problems with the data:

- Presence of more varieties of street types than anticipated. Like **way**, **pike**, **run** etc.
- Abbreviations of street type names in place of the full name. Like **St.** or **St** instead of **Street**, **Rd** instead of **Road**, **Hwy** instead of **Highway** etc.
- Street names in second-level `k` tags pulled from Tiger GPS and divided into segments, in the following format:

```
<tag k="tiger:county" v="Sussex, DE"/>
<tag k="tiger:name_base" v="Harbor"/>
<tag k="tiger:name_type" v="Rd"/>
```

More varieties of street types

To deal with the street types that were not anticipated, I ran the `view_unexpected_street_types.py` which prints a dictionary of the last word of all the `v` attribute of the second level `tag` tag of `way` tags and each instance of their occurrence. The output of the script looks like as follows. (Not the full output)

```
'Ave': set(['E Cleveland Ave', 'Lancaster Ave']),
'Ave.': set(['Lee Ave.']),
'Blvd.': set(['Edwards Blvd.']),
'Rd': set(['Lighthouse Rd',
           'Middletown Warwick Rd',
           'Paddock Rd',
           'Ross Station Rd']),
```

```
  'Rd.': set(['Middleboro Rd.']),
  'East': set(['Mill Landing East']),
```

The varieties of street types that I did not anticipate were added to the `expected_street_types` list of the `config.py` script.

Abbreviations of street type names

As visible in the output above, some street types like Avenue, Road and many more were abbreviated. To replace these with their respective full names the `update_street_types.py` script is run which replaces the abbreviated street type with the full street type name using the `mappings` dictionary.

We have to keep in mind that although some of the last words in the output of the `view_unexpected_street_types.py` are abbreviations, some are true ways of representing a street name like "East" in the above output.

The `update_name` function is the one that replaces the name as per the entries in the `mappings` dictionary. It splits the name of the street using Python's `split()` function and then replacing each of the resultant strings with the corresponding one in the `mappings` dictionary.

```
def  update_name(name, mapping):
        split_name = name.split();
        for i in  range(len(split_name)):
                if split_name[i] in mapping.keys():
                        split_name[i] = mapping[split_name[i]]
        name =  ""
        for sub_name in split_name:
                name += sub_name
                name +=  " "
        name = name[:-1]
        return name
```

The `view_street_types.py` can be run after the above operation to check if we there are any street types that still need to be updated.

Data Overview

The following sections some basic statistics about the data used in this project.

File name	Contents	Size
delaware-latest.osm	Full OpenStreetMap data	171 MB
test_data.osm	Subset of the full data (for testing)	29.9 KB
delaware.db	SQLite database	99 MB
nodes.csv	`<node>` tag data	56.4 MB
nodes_tags.csv	2nd level `<tag>` tag data in `<node>` tag	2.38 MB

File name	Contents	Size
ways.csv	`<way>` tag data	3.44 MB
ways_tags.csv	2nd level `<tag>` tag data in `<way>` tag	15.5 MB
ways_nodes.csv	2nd level `<ref>` tag data in `<way>` tag	27.2 MB

Statistics about users

OpenStreetMap no longer provides data about the user who submitted the data as part of the map data. Consequently, the `user` and `uid` attributes are no longer part of the tags. (Appropriate changes have been made to the related files to deal with this)

Hence, it is not possible to compute any statistics about the users like **number of users**, **top contributor** etc.

Number of nodes

```
sqlite> SELECT COUNT(*) FROM nodes;
996917
```

Number of ways

```
sqlite> SELECT COUNT(*) FROM ways;
101231
```

Top counties in Delaware

```
sqlite> SELECT tags.value, COUNT(*) as count
   ...> FROM (SELECT * FROM nodes_tags UNION ALL
   ...>       SELECT * FROM ways_tags) tags
   ...> WHERE tags.key LIKE '%county_name'
   ...> GROUP BY tags.value
   ...> ORDER BY count DESC LIMIT 10;
```

The results are tabulated for readability:

County	count
Sussex	122
New Castle	77
Kent	54
Salem	6

County	count
Delaware	1
Dorchester	1
Wicomico	1

Top 10 cities in Delaware

```
sqlite> SELECT tags.value, COUNT(*) as count
   ...> FROM (SELECT * FROM nodes_tags UNION ALL
   ...>       SELECT * FROM ways_tags) tags
   ...> WHERE tags.key LIKE '%city'
   ...> GROUP BY tags.value
   ...> ORDER BY count DESC LIMIT 10;
```

The results are tabulated for readability:

City	count
Millsboro	1580
Fenwick Island	889
Newark	587
Bethany Beach	444
South Bethany	407
Rehoboth Beach	385
Georgetown	355
Millville	245
Wilmington	188
Seaford	165

Top 10 appearing amenities

```
sqlite> SELECT value, COUNT(*) as num
   ...> FROM nodes_tags
   ...> WHERE key='amenity'
   ...> GROUP BY value
   ...> ORDER BY num DESC
   ...> LIMIT 10;
```

The results are tabulated for readability:

Amenity	count
restaurant	232
parking	202
bench	87
fountain	55
fast_food	45
place_of_worship	40
toilets	40
fuel	36
school	35
townhall	34

Biggest religion

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
   ...> FROM nodes_tags
   ...>     JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE
value='place_of_worship') i
   ...>     ON nodes_tags.id=i.id
   ...> WHERE nodes_tags.key='religion'
   ...> GROUP BY nodes_tags.value
   ...> ORDER BY num DESC
   ...> LIMIT 1;

christian | 29
```

No surprises here.

Most popular cuisines

```
sqlite> 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
   ...> LIMIT 10;
```

The results are tabulated for readability:

Cuisine	count
pizza	22
mexican	14
american	11
chinese	11
regional	8
ice_cream	6
seafood	6
italian	5
burger	4
chicken;american	4

Pretty sure that "pizza" is not a cuisine but it does seem to be the most popular.

Additional Ideas

Quality of submissions

Unlike in the past, OpenStreetMap no longer provides the details about the user that contributed to the map. Hence, the quality of the data contributed becomes questionable as there is now no way to either incentivize the users submitting accurate map data or restrict users submitting incorrect or low quality data.

We can approach this issue by first investigating the sources from where the map data is predominately collected by looking at the top 10 data sources.

Top 10 data sources

```
sqlite> SELECT value, COUNT(*) as num
   ...> FROM nodes_tags
   ...> WHERE key='source'
   ...> GROUP BY value
   ...> ORDER BY num DESC
   ...> LIMIT 10;
```

The results are tabulated for readability:

Source	count
Bing	5401
Yahoo	415
image	406

Source	count
survey	368
Bing; inferred	271
inferred	127
USGS Geonames	123
county_import_v0.1	91
Bing 2010-2011	76
tiger:boundaries	60

The top source by far is **Bing** which I assume is coming from Bing Maps which is maintained my Microsoft. There is also **Yahoo** at the second place which may come from Yahoo Maps but the third is **image**. What it exactly refers to can be found here in the OpenStreetMap Wiki here. The Wiki says:

> The **image** key has been used by mappers to link an externally hosted image that depicts the tagged object. However, there is an ongoing controversy whether such tags should be part of the OSM database at all.

Now, this is exactly the problem I am trying to propose a solution to, where the quality of the data is questionable.

Proposed solution

In addition to the the contributors, we can have "reviewers". Reviewers are people who can propose edits or updates to the data in their surrounding areas hence acting as local guides to improve the quality of data. Further, we can gamify the process by having reviewer leaderboards and giving benefits to the top reviewers.

Advantages

- The reviewers can only propose changes to the areas near them. Their location can by accessed from their IP address. The amount of area they can propose changes to decreases as the population density in that area increases or they can propose changes to the entire city they reside in.
- Restricting the amount of area reviewers can propose changes to based on the population density can somewhat help with the problem faced in large cities where people living in one area of the city aren't familiar with other regions of the city.
- One beneficial advantage of gamification is that, we can give the "top reviewers" access to a larger area for review as they are more committed to the cause.

Disadvantages

- Obviously, since this approach is again manual, the factor of **human error** always exists.

Conclusion

After the review of the OpenStreetMap data for the state of Delaware it is clear that the map data needs some cleaning to be viable as a consumer mapping service. The data was although cleaned and updated wherever

necessary for the purposes of this project. There is also some concern about the quality of the data contributed by the users with a lot of data coming from sources other than the users like Bing, surveys and Tiger GPS systems. I believe that the data quality can be improved by implementing the above proposed solution to make OpenStreetMap a better mapping service. In spite of these shortcomings, there is lot of insight that can be gained from this data as explored in this project.