OpenStreetMap Data Case Study

Map Area

Savannah, Georgia, United States

https://mapzen.com/data/metro-extracts/#savannah-georgia (https://mapzen.com/data/metro-extracts/#savannah-georgia)

I chose this area because for two reasons. First, it is our home state. But more importantly, my wife's company is constructing a warehouse near the Port of Savannah. This gave us something interesting to discuss over dinner.

Problems in the map data

Due to the relatively small file size, I was able to conduct some basic analysis on the entire file. I discovered the following "questionable items" in the data.

- Extra and missing zip codes: The file contained some zipcodes from Bluffton, SC. While Bluffton is only 20 miles away from downtown Savannah, it is in another state. You also would have to pass through another SC zipcode to reach Bluffton. The file also seemed to be missing zipcode 31407. This notable omission covers two economically important areas, Port Wentworth and the Port of Savannah.
- Inconsistent use of street name abbreviations: (Ave // Avenue, St // Street, etc.) Apparently, both formats are valid for OpenStreetMap. I chose to standardize on the longer form where possible. I did so thinking the primary methods of viewing OpenStreetMap data would be on a computer or mobile device. Having the names listed out would probably make for a more enjoyable viewing experience. The US Postal Service prefers the abbrevated version of the street names, so standardizing on this format would have been acceptable as well.
- Typo on city name: At least one record contained the value "Savannag" for city.
- Inconsistent address format on ways tags: There appears to be at least 3 different formats for address information in the ways tags, further complicating address clean-up efforts.
  - type = "tiger" splits address between 2 lines key = name_base (Fish) and key = name_type (St)
  - type = "regular" has street on one line key = name (Deerfield Road)
  - type = "addr" has street on one line key = street (Government Road).
- Missing roads: My wife's company also owns existing warehouses in the area. I took five of the addresses and went looking for them in the map data. However, none of the addresses appear in the OSM data. The addresses do not appear in the nodes, and the streets do not appear in the ways. This made me wonder if there was a problem with the way I parsed the file and loaded the database. So I double-checked my findings on the OpenStreetMap website. I was actually relieved to see those roads are missing from the website as well. My wife believes this portion of the map data is at least five years out of date.

Zip Code check

In [ ]:
```
sqlite> SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM node_tags
      UNION ALL
      SELECT * FROM way_tags) tags
WHERE tags.key='postcode'
GROUP BY tags.value
ORDER BY count DESC;
```

In [ ]:
```
31401|140
31322|115
29910|8  (<-- Bluffton, SC 20 miles north of Downtown Savannah)
31328|5
31405|5
31302|2
31324|2
31406|2
31408|2  (close to Port of Savannah)(31407 = Port Wentworth, Port of S
avannah)
31410|1
31412|1
31419|1
31406-4805|1

I then cross-referenced this output with the map at http://www.zipmap.
net/Georgia/Chatham_County/Savannah.htm.
```

Cleaning street names

I used the following code to clean up street names. It was adapated from our course work. Again, the "expected" and "mapping" values could just as easily have been reversed. I simply opted for consistency and readability.

In [ ]:
```
########################################
# functions to standardize street names (Street vs St)

street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)

expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place"
, "Square", "Lane", "Road",
            "Trail", "Parkway", "Commons", "Circle", "Expressway", "Wa
lk","Way"]

mapping = { "St": "Street",
            "Blvd": "Boulevard",
            "Ave": "Avenue",
            "Rd": "Road",
            "Cir": "Circle",
            "Ct": "Court",
            "Dr": "Drive",
            "Ln": "Lane",
            "way": "Way"
            }

def update_name(name, mapping):
    # YOUR CODE HERE
    #print "Name: ", name, "// Mapping: ", mapping
    #find the last word
    m = street_type_re.search(name)
    #if there is a last word, store it, and replace it with the mappin
g
    if m:
        street_type = m.group()
        if street_type not in expected:
                name = name.replace(street_type, mapping[street_type])
        #print "M: ", m, "//Street Type: ", street_type
    return name

#######################################
```

Count of cities

In [ ]:
```
sqlite> SELECT tags.value, COUNT(*) as count
   ...> FROM (SELECT * FROM node_tags UNION ALL
   ...>       SELECT * FROM way_tags) tags
   ...> WHERE tags.key LIKE '%city'
   ...> GROUP BY tags.value
   ...> ORDER BY count DESC;
```

In [ ]:
```
Savannah|846
Pooler|165
10|9
Bluffton|9
Tybee Island|5
20|4
Bloomingdale|2
40|1
50|1
425|1
208 MW|1
Daufuskie Island|1
Savannag|1   (<-- error)
```

Missing or incomplete roads

In [ ]:
```
sqlite> select distinct value
   from (select value from way_tags union select value from node_tags)
    where value like 'Gulfs%';
Gulfstream
Gulfstream Aerospace
Gulfstream Aerospace Corp., RDC 1
Gulfstream Aerospace Corp., RDC 2
Gulfstream Aerospace Corp., RDC 3
Gulfstream Aerospace Corp., RDC 4
Gulfstream Aerospace Corp., RDC Labs
Gulfstream Road
Gulfstream Service Center

(Note: Gulfstream Road _is_ a desired result, but the portion where th
e warehouses does not seem to be in the data.)
```

In [ ]:
```
sqlite> select distinct value
    from (select value from way_tags union select value from node_tags
)
    where value like 'Logis%';
sqlite>
sqlite> select distinct value
    from (select value from way_tags union select value from node_tags
)
    where value like 'Expan%';
sqlite>
```

Data Overview

Here are some basic statistics on the dataset, the SQL queries used to gather them, and the output of those queries.

File Sizes

In [ ]:
```
savannah_georgia.osm ....... 72 MB
sav1.db ................... 41.1 MB
nodes.csv ................. 28.1 MB
nodes_tags.csv ............ 685 KB
ways.csv .................. 2 MB
ways_tags.csv ............. 4.1 MB
ways_nodes.csv ............ 9.5 MB
```

Number of Nodes

In [ ]:
```
sqlite> select count(*) from node;

338652
```

Number of Ways

In [ ]:
```
sqlite> SELECT COUNT(*) FROM way;

32909
```

Number of unique users

In [ ]:
```
sqlite> select count(distinct(e.uid)) from (select UID from node union all select UID from way) e;

306
```

Top 10 contributing users

In [ ]:
```
sqlite> select e.user, count(*) as num
   ...> from (select user from node union all select user from way) e
   ...> group by e.user
   ...> order by num desc
   ...> limit 10;
```

In [ ]:
```
hokieengr ....... 119260 (32.1% of nodes + ways)
MikeNBulk ....... 67362  (18.1%)
woodpeck_fixbot . 32833  (8.8%)
EdLoach ......... 14098  (3.8%)
Rub21 ........... 14045  (3.8%)
ediyes .......... 14044  (3.8%)
maven149 ........ 11897  (3.2%)
coleman ......... 9259   (2.5%)
Liber ........... 7497   (2.0%)
dmgroom_ct ...... 6001   (1.6%)

Number of Nodes + Number of Ways = 371,561
Total percentage from top 2 users: 50.2%
Total percentage contributions from top 10 users: 79.7%
```

Number of users appearing only once

In [ ]:
```
sqlite> select count(*)
   ...> from
   ...> (select e.user, count(*) as num
   ...> from (select user from node union all select user from way) e
   ...> group by e.user
   ...> having num = 1) u;

72 (23.5% of unique users)
```

Both the top two and top ten users contributed a lower percentage of submissions than I expected. (50.2% and 79.7%, respectfully) While these may still be higher than desired, even within the top ten the percentages start to trail off significantly.

Additional Ideas

I'm primarily concerned about the missing data and potentially the age of the map data. One easy way to gather more updated data points is to work with my wife's company. There's only a handful of people who are working on the projects. It would be relatively easy to get them to drive the area to collect updated data automatically. We could also just gather in a conference room and review the OSM website over a free lunch. We could perform simple edits right there, or brainstorm on a more comprehensive approach to upload fresh data.

Another way to improve the map data for Savannah would be to work with local trucking companies. There are several trucking companies servicing the port area. They primarily run short routes between the ports and local warehouses. We could identify one company and run a contest to see which driver can provide the most updates to the map data. We could also run the contest between trucking companies if the need exists and another company could be identified. The winners would receive bragging rights and dinner at a restaurant of their choice.

Additional Data Exploration

Top 10 appearing amenities

In [ ]:
```
sqlite> select value, count(*) as num
   ...> from node_tags
   ...> where key = 'amenity'
   ...> group by value
   ...> order by num desc
   ...> limit 10;
```

In [ ]:
```
place_of_worship ..... 336
school ............... 125
restaurant ........... 80
grave_yard ........... 49
bench ................ 46
fast_food ............ 26
parking .............. 24
library .............. 17
hospital ............. 16
fuel ................. 15
```

Biggest religion

In [ ]:
```
sqlite> SELECT node_tags.value, COUNT(*) as num
   ...> FROM node_tags
   ...> JOIN (SELECT DISTINCT(id) FROM node_tags WHERE value='place_of
_worship') i
   ...> ON node_tags.id=i.id
   ...> WHERE node_tags.key='religion'
   ...> GROUP BY node_tags.value
   ...> ORDER BY num DESC
   ...> LIMIT 1;
```

In [ ]:
```
christian ......... 320
```

Top Restaurant types

In [ ]:
```
sqlite> SELECT node_tags.value, COUNT(*) as num
   ...> FROM node_tags
   ...>     JOIN (SELECT DISTINCT(id) FROM node_tags WHERE value='rest
aurant') i
   ...>     ON node_tags.id=i.id
   ...> WHERE node_tags.key='cuisine'
   ...> GROUP BY node_tags.value
   ...> ORDER BY num DESC;
```

In [ ]:
```
american .. 7
burger .... 5
mexican ... 5
pizza ..... 3
seafood ... 3
french .... 2
japanese .. 2
regional .. 2
sandwich .. 2
asian ..... 1
burrito ... 1
chinese ... 1
cookie .... 1
diner ..... 1
greek ..... 1
italian ... 1
noodle .... 1
```

Conclusion

Working with the Savannah data was very interesting. Even though I have no formal ties to the area, it was fun to learn how to parse the map data and clean some of it programatically. This could be used as a starting point to refresh and update some of the data in the area. Given the amount of missing, incomplete, and outdated data I quickly discovered, I believe it would be easy to dramatically improve the quality of the data in just a few hours.