OpenStreetMap Data Project Report

By Logan Burke

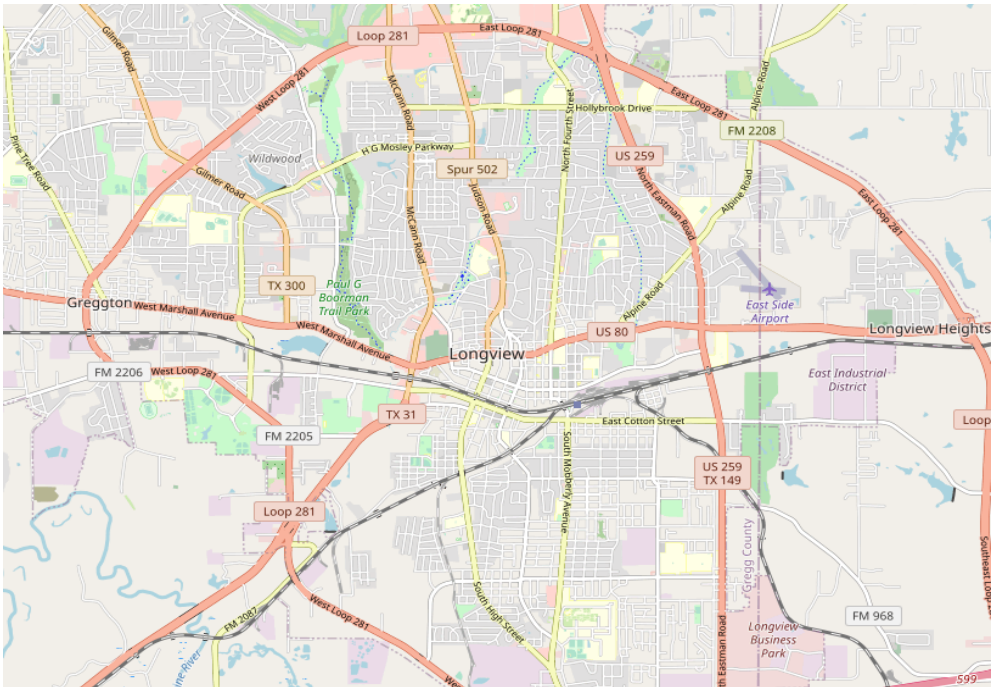
4/23/2021

In this report, I will be looking at data from the OpenStreetMaps website that I downloaded and processed for this project.

Map Area Used

Longview, TX, United States

- Link for OpenStreetMap data (this is the sample of Longview[size is 8.31 MBs): https://www.openstreetmap.org/export#map=15/32.5258/-94.7251
- Link for OpenStreetMap data (this is the full area and is 85.1 MBs in size[Downloaded using Overpass API]): https://www.openstreetmap.org/export#map=11/32.5257/-94.7252



This area is where I live now and I am interested in knowing all the interesting facts that I can find with my queries for my new home.

Problems Encountered in the Map

After initially downloading the file, I ran some processing code so that I could get an idea of what I was dealing with. I created a database file and ran some exploration sql code to see what I could

find. I noticed some of the following problems with the data entered.

- Incorrectly entered street names (“H G Mosley” instead of “H.G. Mosley”)
- Inconsistent phone numbers("+1 903 236 7468","+19032386928","+1-903-753-2773")
- Inconsistent entries under amenities (“place_of_worship”, “grave_yard”)
- Amenity tag typo (“community_centre” instead of “community center”)
- Barrier tag having (“cattle_grid”, “swing_gate” without proper spacing)
- Cuisine tag issues (“stuff” needs to be fixed. “Ice_cream”, also several like this “american;burger;seafood;steak;italian;mexican;dessert;bar”)

Abbreviated Street Names

To clean the abbreviated street names I used dictionaries for look-ups. The function 'update_addr_name' is implemented in file clean.py. It takes in the street name as input and returns the cleaned street name. Here is a sample of the output.

St ---> Street
St. ---> Street
Ave ---> Avenue
Rd ---> Road
Road ---> Road
road ---> Road
cross ---> Cross
ROad ---> Road
ROAD ---> Road
Dr ---> Drive
Blvd ---> Boulevard
Ct ---> Court
Ln ---> Lane
Hwy ---> Highway
Cir ---> Circle
Pky ---> Parkway
Trl ---> Trail
Pl ---> Place
Sq ---> Square
Cv ---> Cove
Trce ---> Terrace

I also had it check and correct for any directional address (“North”, “South”, etc.) and also if it was a numbered street (“1st”, “2nd”, etc.) to normalize the data. Here is a sample of that output.

Directional abbreviations:
N ---> North
S ---> South
E ---> East
W ---> West

N. ---> North
S. ---> South
E. ---> East
W. ---> West

Numbering abbreviations:
1st ---> First
2nd ---> Second
3rd ---> Third
4th ---> Forth
5st ---> Fifth
6th ---> Sixth
7th ---> Seventh
8th ---> Eighth
9th ---> Ninth
10th ---> Tenth
11th ---> Eleventh
12th ---> Twelfth
13th ---> Thirteenth
14th ---> Fourteenth
15th ---> Fifteenth

Inconsistent City Names

To clean the city names I used a dictionary 'mapping_city' for look-ups to match city names to corrected names. The function 'update_city_name' is implemented in file cleaning.py. Here is an example of that output.

City name corrections:
longview ---> Longview
long view ---> Longview
Long view ---> Longview
Long View ---> Longview
lakeport ---> Lakeport
hallsville ---> Hallsville
kilgore ---> Kilgore
WhiteOak ---> White Oak
whiteoak ---> White Oak
white oak ---> White Oak
diana ---> Diana
gilmer ---> Gilmer

Speed Limit Inconsistencies

Some of the speed limits were missing the “mph” and so I wrote some code to correct this.This function 'speedlimit' is implemented in cleaning.py

Inconsistent Phone Numbers

The phone numbers were in a couple of different formatting options and I made them into a standard format: (xxx) xxx-xxxx. I used regex to get all the phone numbers which were in standard form. The rest of the phone numbers were parsed with extra +1,'.' etc removed and made into standard form. This function 'update_phonenum' is implemented in the cleaning.py file.

Phone number correction example:
+1-903-986-5500 ----> (903) 986-5500
+19037598545 ----> (903) 759-8545
+1 903 297 8030 ----> (903) 297-8030

Data Overview

The file sizes for each of the files are as follows:

* Longview_Full.osm---------- 85.1 MB
* Longview_Full.db---------- 43.7 MB
* nodes.csv---------- 33.2 MB
* nodes_tags.csv---------- 337 KB
* ways.csv---------- 2.5 MB
* ways_tags.csv---------- 4.6 MB
* ways_nodes.csv---------- 11.2 MB

Number of Nodes

SQL: SELECT count(*) FROM nodes;
OUTPUT: 410033

Number of Nodes_Tags

SQL: SELECT count(*) FROM nodes_tags;
OUTPUT: 9472

Number of Ways

SQL: SELECT count(*) FROM ways;
OUTPUT: 43183

Number of Ways_Tags

SQL: SELECT count(*) FROM ways_tags;
OUTPUT: 138891

Number of Unique Users

SQL: SELECT count(DISTINCT(users.uid)) FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) users;
OUTPUT: 284

Top 10 Contributors

SQL: SELECT users.user, count(*) as num FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) users GROUP BY users.user ORDER BY num DESC LIMIT 15;

OUTPUT:
(User / Contributions)

TAMU2016	400280
woodpeck_fixbot	7987
stangoodman	5285
25or6to4	5047
karlcr9911	4553
TexasNHD	3428
GB1919	3338
unsungNovelty	2297
swimdb	1338
kbulgrien	767
ElliottPlack	705
clay_c	668
pzharsh	541
mzamam	526
greggerm	443

Number of users having 1 post

SQL: SELECT COUNT(*) FROM (SELECT users.user,COUNT(*) as num FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) users GROUP BY users.user HAVING num=1);

OUTPUT: 35

Additional Data Exploration

Top 10 Cuisines

SQL: SELECT value,count(*) as cnt FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) WHERE key == ‘cuisine’ GROUP BY value ORDER BY cnt DESC LIMIT 10;

OUTPUT:

Burger	40
mexican	18
sandwich	13
american	11
Pizza	10
chicken	10

Tex-mex	8
seafood	6
chinese	5
Ice_cream;burger	3

Top 10 Leisure Amenities

SQL: SELECT value,count(*) as cnt FROM ways_tags WHERE key == ‘leisure’ GROUP BY value ORDER BY cnt DESC LIMIT 10;

OUTPUT:

pitch	238
playground	90
swimming_pool	56
park	42
bleachers	33
sports_centre	14
golf_course	8
garden	8
fitness_centre	6
stadium	5

Top Religions

SQL: SELECT value,count(*) as cnt FROM ways_tags WHERE key == ‘religion’ GROUP BY value ORDER BY cnt DESC;

OUTPUT:

christian	102
jewish	1

Places of Worship

SQL: SELECT DISTINCT value,count(*) as cnt FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) WHERE key == 'amenity' GROUP BY value Order by cnt DESC;

OUTPUT: 170

City names

SQL: SELECT DISTINCT value FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) WHERE key == 'city' and type == 'addr';

OUTPUT:

'Longview'
'Kilgore'

'Hallsville'
'White Oak'
'Lakeport'
'Gilmer'
'Diana'

Additional Suggestions

It would be great to include more accessibility information like if a location is wheelchair accessible, or historic sites and farmer's markets information. It would also be great if geocaching was listed.

To improve the quality of the data, I suggest that there should be a standard format that the contributors can enter the data or the system should reject it. Also requiring all data be entered would be great too like if someone entered a 'park' they would need to add such information like 'public bathroom', 'wheelchair accessible', 'park curfew'. Etc. This would really help flesh out the information and give more useful data. Also some of the tags are very nebulous and could use more descriptive titles ('name', name_base', 'zipcode_left', 'zipcode_right', etc.). This could make contributors less willing to enter information, but we could simply have default filler tags of "none". We could also encourage the contributor by displaying their rank as a contributor as compared to others who have entered data, gamifying the process somewhat. The contributor may be an end user of the map as well and would find it much easier to retrieve error-free data from the map.

Conclusion

The OpenStreetMap of Longview Texas is of reasonably good quality that has an immense amount of useful information but has some minor issues that hold it back such as typos and different formats of the same data as well as some nebulous and repeating data.

References

- Link for OpenStreetMap data (this is the sample of Longview[size is 8.31 MBs): https://www.openstreetmap.org/export#map=15/32.5258/-94.7251
- Link for OpenStreetMap data (this is the full area and is 85.1 MBs in size[Downloaded using Overpass API]): https://www.openstreetmap.org/export#map=11/32.5257/-94.7252
- Stack overflow for various python questions http://stackoverflow.com/
- Reg expressions regexone.com/references/python
- Stack overflow for various SQL questions http://stackoverflow.com/
- Python.org website for information on sqlite https://docs.python.org/3/library/sqlite3.html