OpenStreetMap Data Project

author: Bob Cross

date: December 15, 2016

Map Area

I chose Austin, Texas as the area for this project. I am new to this city and thought this project would assist me in exploring Austin.

Identifying Problems in the Map

After taking a sample of the 1.16 GB dataset using sample_region.py, I used three techniques to identify problems in the sample:

Utilized Sublime to view portions of the data in it's original form.
Analyzed the audit.py script output to view unusual street names and postal codes.
Analyzed the CSV files created by the process_osm. script to view the data (in schema.md format) before and after cleaning code was applied.

Problems Encountered in the Map

Simplified versions of code for audit and cleaning the following problems is presented below.

Overabbreviated street names

Spell out over abbreviated street tyoes:

> import xml.etree.cElementTree as ET from collections import defaultdict import re import pprint

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

expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane", "Road", "Trail", "Parkway", "Commons"]

UPDATE THIS VARIABLE mapping = { "St": "Street", "St.": "Street", "Ave.": "Avenue", "Rd.": "Road" }

def audit_street_type(street_types, street_name): m = street_type_re.search(street_name) if m: street_type = m.group() if street_type not in expected: street_types[street_type].add(street_name)

def is_street_name(elem): return (elem.attrib['k'] == "addr:street")

def audit(osmfile): osm_file = open(osmfile, "r") street_types = defaultdict(set) for event, elem in ET.iterparse(osm_file, events=("start",)):

```
        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    osm_file.close()
    pprint.pprint (dict(street_types))
    return street_types
```

def update_name(name, mapping):

```
    # YOUR CODE HERE
    m = street_type_re.search(name)
    if m:
        street_type = m.group()
        if street_type in mapping.keys():
            print 'Before: ', name
            name = re.sub(m.group(), mapping[m.group()], name)
            print 'After: ', name


    return name
```

Erroneous City names in dataset

```
import xml.etree.cElementTree as ET import pprint
```

audit way tags for unexpected city names in Austin map

osmfile = "sample.osm"

expected = ("Austin", "Pflugerville", "Round Rock")

def is_city_name(elem): return (elem.attrib['k'] =="addr:city")

def audit_city(osmfile): city_file = open(osmfile, "r") for event, elem in ET.iterparse(city_file, events=("start",)): if elem.tag == "node" or elem.tag == "way": for tag in elem.iter("tag"): if tag.attrib['k'] == 'addr:city': city = tag.attrib['v'].strip() if city not in expected: print city

```
city_file.close()
```

audit_city("sample.osm")

audit of sample file shows names which appear to be outside of what would be considered Austin (eg Buda > 60mi away)

this would require edit specific entries; these can be ignore for analysis purposes in most cases with no material effect

```
Manchaca Cedar Park, TX Buda Sunset Valley Buda Lakeway Sunset Valley Cedar Park
```

Postal Codes

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

abbreviated results of unusual values:
Texas|2
78724-1199|10
78728-1275|2
78754-5701|2
78759-3504|2
78704-5639|1
78704-7205|1
78753-4150|1
78758-7008|1
78758-7013|1
TX 78613|1
TX 78724|1
TX 78728|1
TX 78735|1
TX 78745|1
TX 78758|1

removal of the items listed as "Texas"; "TX" pre-fix and four digit extensions can be ignored in analysis

Data Overview

File sizes

austin_map.osm .... 1.16 GB
austin_map.db .... 773 MB
nodes.csv .... 499 MB
nodes_tags.csv .... 9 MB
ways.csv .... 40 MB
ways_tags.csv .... 144 MB
ways_nodes.cv .... 59 MB

for validation and data familiarity:
austin_map_sample.osm .... 59 MB
test_sample.osm .... 9 MB

Number of nodes

sqlite> SELECT COUNT(*) FROM nodes;
5,358,645

sqlite> SELECT COUNT(*) FROM nodes_tags;
2,011,149

Number of ways, ways_nodes and ways_tags

sqlite> SELECT COUNT(*) FROM ways;
564388

sqlite> SELECT COUNT(*) FROM ways_nodes;
58,992,639

SELECT COUNT(*) FROM ways_nodes;
5,892,639

Number of unique users

sqlite> SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
1644

Top 10 contributing users

sqlite> 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;

patisilva_atxbuildings|2492624
ccjjmartin_atxbuildings|1062000
ccjjmartin__atxbuildings|834577
jseppi_atxbuildings|273706
wilsaj_atxbuildings|249835
kkt_atxbuildings|157845
lyzidiamond_atxbuildings|135318
woodpeck_fixbot|77816
johnclary_axtbuildings|48232
richlv|46822

Number of users with only 1 posting

sqlite> 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;
238

Additional Data Exploration

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;
```

```
parking|1885
restaurant|700
waste_basket|601
fast_food|501
school|437
place_of_worship|413
bench|359
fuel|347
shelter|230
bank|150
```

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;
mexican|60
american|27
pizza|23
chinese|22
indian|14
sandwich|14
italian|13
regional|13
sushi|13
thai|13
```

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|367
```

Dataset Improvement

Classification issues from original source input is one of the most significant issues with this dataset and one of the most challenging to address. This is most often the cases with new contributors to the dataset. A periodic scrubb of input from contributors with less than 5 entries may effectively address this concern.

Conclusion

The Austin OpenStreetMap dataset is a quite large and a bit messy. However, I believe the dataset is not materially deficint for the analysis purposes of this project. By quwrying the dataset, I learned a number of new things about Austin which will benefit me as I continue to explore this town. The dataset is very useful, though areas for improvement exist.