OpenStreetMap Data Project

author: Bob Cross
date: December 15, 2016
re-submitted December 21, 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.

Over abbreviated street names

Audit and cleanig code and output:

```
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):
    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

def test():
    st_types = audit(OSMFILE)
    #pprint.pprint(dict(st_types))

    for st_type, ways in st_types.iteritems():
        for name in ways:
            better_name = update_name(name, mapping)
            #print name, "=>", better_name
  test()
```

Partial output below:
Before: Woodrow Ave.
After: Woodrow Avenue
Before: Pecan St.
After: Pecan Street
Before: E 38th 1/2 St.
After: E 38th 1/2 Street
Before: E. 43rd St.
After: E. 43rd Street
Before: Pecan St
After: Pecan Street
Before: Rio Grande St
After: Rio Grande Street

Postcodes with prefix and suffix and other issues

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

osmfile = "sample.osm"

postcodes = defaultdict(set)

mapping = {'78724-1199' : '78724',
          'TX 78745' : '78745'
          }

def is_postcode(elem):
    return (elem.attrib['k'] == "addr:postcode")

def audit_postcode(postcodes, postcode):
    expected = (map(str,range(78610,78799)))
    if postcode not in expected:
        postcodes[postcode].add(postcode)
```

```
    return postcodes

def update_postcode(postcode):
    if postcode in mapping.keys():
            print 'Before: ', postcode
            postcode = mapping[postcode]
            print 'After: ', postcode
    return postcode



def audit(osmfile):
    osm_file = open(osmfile, "r")

    postcodes = defaultdict(set)
    #for i, elem in enumerate(get_element(osmfile)):
    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_postcode(tag):
                    audit_postcode(postcodes, tag.attrib['v'])
    pprint.pprint(dict(postcodes))

    osm_file.close()

    return dict(postcodes)
    #pprint(dict(postcodes))

def test():
    flagged_postcodes = audit(osmfile)
    #print flagged_postcodes



    for key in flagged_postcodes:
            update_postcode(key)



test()
```

```
{'78724-1199': set(['78724-1199']), 'TX 78745': set(['TX 78745'])}
Before:  TX 78745
After:  78745
Before:  78724-1199
After:  78724
```

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 scrub of input from contributors with less than 5 entries may effectively address this concern. Additionally and in a similar category, there are manual input errors which could be flagged and with the periodic scrub be corrected. For example in running the postcode audit, I noted two postcodes which were entered as “TEXAS”; during the scrub the lat/lon values could be compared in the Python library “uszipcode” utilizing the 'ZipcodeSearchEngine' and then be properly corrected.

Marking the Open Street Maps with more handicap accessibility would information would add value to the database. There were only 194 nodes_tags marked for wheelchair accessibility.

```
SELECT COUNT(*) FROM nodes_tags WHERE key='wheelchair';
194
```

This effort could be assisted by reference to similar mapping sites and governmental guidelines such as the links listed below:

Disabled Accessibility – axsmap.com and **Guidelines for ADA and Texas Accessibility Standards - Adaptive Access**

Accessibility information for hundreds of restaurants, cafes, tourist attractions, community centers, and other public spaces could be added to the dataset. Programmatically extracting the yes/no information and adding it to the OpenStreetMap dataset would likely be most efficient. One difficulty would be dealing with naming inconsistencies between the reference resources data and the nodes already in the OpenStreetMap dataset. This could be overcome with careful string handling and a careful scrub of the input data.

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.