

OpenStreetMap Data Case Study

Map Area

Tampa, Florida, United States

I chose Tampa, Florida because it is my current hometown and I am curious to see the what could be learned from the data. Also, being familiar with the area I hope that my local knowledge would help in the data cleaning process and ultimately contribute to the OpenStreetMap project.

Issues with the Data

1. Inconsistent abbreviations for street names.
A. ex: Street/St., Avenue/Ave, Trail/Trl
2. Inconsistent postal code formats
A. ex: 33613-4649; 33701:33704

Inconsistent Abbreviations

Running an initial audit on the street names revealed an inconsistent use of abbreviations. Some locations would use the full spelling others would use abbreviated versions. (Street/St./St)

Here I opted to iterate over each and corrected the abbreviations to the full spelling using the `update_name()` function and a mapping dictionary as follows:

```

In [1]: mapping = { "St": "Street",
                    "St.": "Street",
                    "Ave" : "Avenue",
                    "Ave.": "Avenue",
                    "Rd.": "Road",
                    "Rd": "Road",
                    "E" : "East",
                    "W" : "West",
                    "S" : "South",
                    "N" : "North"
                    }

def update_name(street_name, mapping):
    street_name = street_name.replace(' ', ' ')
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        street_type2 = ' '.join(street_name.split()[0:])
        if street_type in mapping.keys():
            #print 'Before: ', name
            street_name = re.sub(street_type,
mapping[street_type],street_name)
            #print 'After: ', name

    return street_name

```

This will replace any abbreviated street name strings to the full name and adding consistency to the data.

Postal Codes

Postal code strings again needed to be addressed for their inconsistency. Here again I iterated over each string and updated any string that was over 5 digits as follows:

```

In [ ]: def update_postal_code(postal_code):
        postal_code = postal_code.upper()
        if ' ' not in postal_code:
            if len(postal_code) != 5:
                postal_code = postal_code[0:5]
        return postal_code

```

This updated all problematic zip codes turning "33613-4649" into "33613".

Data Overview

This section contains exploratory statistics and the SQL queries implemented.

File sizes

```
tampa_florida.osm ..... 945 MB
tampa_florida.db ..... 138 MB
nodes.csv ..... 150 MB
nodes_tags.csv ..... 7 MB
ways.csv ..... .001 MB
ways_tags.csv ..... .001 MB
ways_nodes.csv ..... .001 MB
```

Number of nodes

```
def nodes_count():
    query = "select count(*) from nodes"
    cursor.execute(query)
    results = cursor.fetchall()
    pprint(results)
```

1,804,986

Number of unique users

```
def user_count():
    query = "select count(sub.uid) from(select uid from nodes union select uid from way
s) sub"
    cursor.execute(query)
    results = cursor.fetchall()
    pprint(results)
```

1,307

Top 10 contributing users

```
def top_users():
    df = pd.read_sql_query("Select user, COUNT(user) as num FROM nodes GROUP BY uid ORD
ER by num DESC;", conn)

    print df[0:10]
```

	user	num
0	woodpeck_fixbot	233459
1	coleman	198403
2	grouper	177396
3	Andrew Matheny_import	162155
4	EdHillsman	90081
5	NE2	66349
6	LnXNoob	54292
7	David Hey	53521
8	KalininOV	52415
9	westampa	40941

Top user contributions

```
df = pd.read_sql_query("SELECT ways_tags.key, COUNT(ways_tags.key)
as num FROM ways_tags JOIN ways ON ways_tags.id = ways.id
WHERE ways.uid = '561234' GROUP BY ways_tags.key ORDER BY num DESC;", conn)
print df
```

Top 10 Amenities

```
def amenities_count():
df = pd.read_sql_query("select value, count(*) as num from nodes_tags where key =
'amenity' group by value \
order by num desc limit 10;", conn)
print df
```

	value	num
0	restaurant	827
1	place_of_worship	749
2	school	500
3	fast_food	375
4	bicycle_parking	353
5	bench	279
6	fuel	217
7	fountain	203
8	bank	157
9	toilets	146

Most popular cuisines

american	90
pizza	70
mexican	41
italian	26
chinese	24
seafood	23
greek	19
sandwich	18
thai	18
burger	16

Suggested Improvements

Problem: Despite having a webpage dedicated to addressing editing standards(https://wiki.openstreetmap.org/wiki/Editing_Standards_and_Conventions (https://wiki.openstreetmap.org/wiki/Editing_Standards_and_Conventions)), the OpenStreet Map project is riddled with inconsistencies in the data. For instance the web page specifically asks for users to not enter in use any abbreviations. However, as we saw above using our `audit_street_names` and `update_names` functions there was clearly a need to address and clean up this part of the data. The same was true for our postal codes. From this we can assume that contributors failed to use the suggested conventions in other aspects of the project.

Possible Solution:

A data entry solution could be a rejection message if nonconventional or unsuggested data is entered. This would help catch possible errors on the front-end and save time but not having to subsequently clean the already submitted data.

