OpenStreetMap Data Case Study

Map Area

New York City + North East Jersey

- https://mapzen.com/data/metro-extracts/your-extracts/f95fe4153f07 (https://mapzen.com/data/metro-extracts/your-extracts/f95fe4153f07)

This is where I grew up and lived, so I thought I would have better intuition on the OpenStreetMap (OSM) data than researching a place I haven't lived. It also sounded fun for me to look at my city through data and make contributions to it.

Problems Encountered in the Map

After taking a look at a smaller subset of the data I noticed there were some rooms for improvement. Here are some of them I decided to fix/improve:

- Over-abbreviated Street Names (ex: '43rd Rd' as opposed to "43rd Road')
- Inconsistent Capitalization of Street Names (ex: 'avenue', Avenue', 'street', 'Street')
- Typos in Street Names (ex: 'Aveneu', 'Avene', 'Steet')
- Inconsistent Postcode Format
  - Some but not all postcodes have "NY" attached to them (ex:'NY 11106').
  - Some but not all postcodes have four-digit extension attached to them (ex: '10011-6832').

Correcting Street Names

To correct street names, I created a dictionary called "mapping", which holds problematic names as its keys and proper street names as its values. Then I iterated over street names in the original data and saw if any street name matched the keys in the dictionary. If there was a match, I changed the street name to the corresponding value in the dictionary. Doing this corrected over-abbreviated street names and inconsistent street names.

Here's a snippet of code that I used to clean aforementioned problems with street names:

In [2]:

```
mapping = {
"Rd": "Road",
"W": "West",
"E": "East",
"S": "South",
"N": "North",
"W ": "West",
"E ": "East",
"S ": "South",
"N ": "Norht",
" W": "West",
" E": "East",
" S": "South",
" N": "Norht",
"avenue" : "Avenue",
"Ave": "Avenue",
"Ave.": "Avenue",
"Aveneu": "Avenue",
"Avene": "Avenue",
"street": "Street",
"ST": "Street",
"St.": "Street",
"St" : "Street",
"st" : "Street",
"Steet" : "Street"
}

def update_name(osmfile):
    osm_file = open(osmfile, "r+")
    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):
                    name = tag.attrib['v']
                    #1 Find if the street name ends with any of the keys in "map
ping",
                    m = street_type_re.search(name) # Pulling out the endname of
the streetname
                    if m:
                        street_type = m.group() # Pulled out endnames of the str
eets
                        if street_type in mapping.keys():
                            #2 If it does, the endname should be changed to the
key's value
                            tag.attrib['v'] = name.replace(street_type, mapping[
street_type])
    osm_file.close()
```

Correcting Postcodes

To standardize inconsistent postcodes, I decided to leave only the 5-digit postcodes. In other words, I got rid of any extension that is attached to the postcodes (ex: "NY", "-0111").

Here is the code that I used to clean the postcodes:

In [ ]:

```
#regular expression that will find 5 digit postcode
fiveDigitPostcode = re.compile('[0-9]{5}', re.IGNORECASE)

def changePostcode(osmfile):
    osm_file = open(osmfile, "r+")
    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):
                    postcode = tag.attrib['v']
                    match = fiveDigitPostcode.search(postcode)
                    if match is not None:
                        #change the postcode to just a 5 digit number.
                        tag.attrib['v'] = match.group()
    osm_file.close()
```

Examining the Data through SQL Querying

After cleaning was done, the data - written in XML - was converted to csv format. It was then imported into a local SQLite3 Database.

Data Overview

File Sizes:

- Entire OpenStreetMap (OSM) Data – 345.5 MB
- nodes.csv – 118.7 MB
- nodes_tags.csv – 8.1 MB
- ways.csv – 15.7 MB
- ways_nodes.csv -43.8 MB
- ways_tags.csv – 41.5 MB
- nyc.db – 208.4 MB

Number of Nodes:

```
sqlite>
SELECT count(distinct(uid))
FROM
(SELECT uid FROM nodes UNION ALL SELECT uid FROM ways);
```

2049

Number of Ways:

```
sqlite> SELECT count(id) FROM nodes;
```

1292949

Number of Unique Users:

```
sqlite> SELECT count(id) FROM ways;
```

234103

Top Ten Most Abundant Amenities:

```
sqlite>
SELECT nodes_tags.value, count()
FROM nodes_tags
WHERE key = 'amenity'
GROUP BY nodes_tags.value
ORDER BY count() DESC
LIMIT 10;
```

bicycle_parking | 3452
restaurant | 1814
cafe | 669
fast_food | 487
school | 454
bar | 338
bicycle_rental | 307
bank | 306
place_of_worship | 295
drinking_water | 287

Additional Statistics:

Most Popular Attraction and Its Prevalence :

```
sqlite>
SELECT nodes_tags.value, count(*) as num
FROM nodes_tags
JOIN (select DISTINCT(id) FROM nodes_tags where key = 'tourism') i
ON nodes_tags.id = i.id
WHERE nodes_tags.key = 'attraction'
GROUP BY nodes_tags.value
ORDER BY NUM DESC;
```

animal | 9

Top 10 Most Prevalent Fast Food Chains:

```
sqlite>
SELECT nodes_tags.value, count()
FROM nodes_tags
JOIN (select DISTINCT(id) FROM nodes_tags WHERE value = 'fast_food') i
ON nodes_tags.id = i.id
WHERE nodes_tags.key = 'name'
GROUP BY nodes_tags.value
ORDER BY count() DESC
LIMIT 10;
```

McDonald's | 49
Subway | 37
Dunkin' Donuts | 17
Chipotle | 11
Chipotle Mexican Grill | 8
Burger King | 7
Shake Shack | 7
Five Guys | 6
Pret A Manger | 6
Taco Bell | 6

Additional Idea for Improving OSM Data

I noticed that some node tags values are written in languages other than English. It would be nice if these could be translated into English so that all users of OSM can recognize what these nodes are.

Here are what some of these node tags values look like:

sqlite>
SELECT value, count() *FROM nodes_tags*
GROUP BY value
ORDER BY count() ASC;

裕利 | 1
長旺飯店 | 1
뉴욕 | 1
록펠러센터 | 1
브라이언트 파크 | 1
न्यूयोर्क् | 1
நியூ யோர்க் | 1
న్యూయార్క్ | 1
ನ್ಯೂಯಾರ್ಕ್ | 1
отель | 1
отель yotel | 1
отель Пенсильвания в Нью Йорке | 1

It would be nice if these could be translated into English so that all users of OSM can recognize what these nodes are. Doing that, however, might take a lot of time and resources as programatically correcting each of these names seems difficult. This is so because there are many different languages in the data and I need inputs from native speakers to make correct translation. Also, some of these nodes might represent sign boards in the buildings that are actually written in the languages we see above. Changing these names to English then might lead to loss of valuable information.

To figure out what some of these node tags stand for, I ran a few queries that looks like this:

sqlite>
SELECT * FROM nodes_tags
WHERE value = 'Таймс-сквер'

299046361|ru|Таймс-сквер|name

The result I found was that a lot of these node tags were hard to understand as meanings of both the key and the value were unclear. Knowing Korean, however, I was able to identify that few of the node tags meant something like "New York", "Rockfeller Center", and "Bryant Park."

Conclusion

Having audited the data, it is clear to me that the New York area has rooms for improvement. Implementing the changes I made will certainly improve parts of it, although not all. Some of the changes will be more difficult to implement than others for reasons I mentioned in the section directly above. Overall, it was a pleasureable experience to audit my city through data.