wrangle_data

October 7, 2017

1 Wrangle OpenStreetMap Data

1.1 Map Area

New York, United States

`https://mapzen.com/data/metro-extracts/metro/new-york_new-york/`

```
I went to New York for study several year agao. I'm interested in exploring the data to see hou
```

1.2 Problems Encountered in the Map

After subseting the small portion data of New York area, and running it against audting code, I noticed that :

1. street type names are not consistent, Avenue has several types, like

   ```
   'Ave': '10th Ave'
   'Ave,': 'Franklin Ave'
   'Ave.': '468 Piermont Ave.'
   'Avene': '8th Avene'
   'Aveneu': 'St. Nicholas Aveneu'
   'Avenue,': '70th Avenue,'
   ```

2. Some of the records doesn't belong to New York area:

   ```
   <tag k="addr:state" v="NJ" />
   <tag k="addr:street" v="Elizabeth Street" />
   <tag k="addr:postcode" v="07735" />
   ```

3. Zip Codes are in defferent formats and having some value not reasonable:

   ```
   11221
   NY 11221
   08854-8003

   11201;11231: this seems to be two zip codes
   115422: longer than zip code
   7011: shorter than zip code
   ```

1.2.1 Fixing Unexpected Street Types

In [1]:
```
"""
Fixing unexpected street types to the appropriate ones in the expected list.

The update_name function fixes the street name. The function takes a string
with street name as an argument and would return the fixed name
"""

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

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


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

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(OSM_PATH):
    osm_file = open(OSM_PATH, "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()
```

```
            return street_types


        def update_name(name, mapping):
            m = street_type_re.search(name)
            if m:
                street_type = m.group()
                if street_type not in expected:
                    name = re.sub(street_type_re, mapping[street_type], name)

            return name
```

1.2.2 Fixing Zip Code Issue

```
In [24]: """
         The function update_postcode is aimed at updating valid zip code values
         which are recored in 'wrong' format
         For zip code values which are invalid, it is ignored by this function
         """

         def update_zipcode(postcode):
             if len(postcode) > 5:
                 post_code = postcode.split()
                 n = len(post_code)
                 if n == 2:
                     return post_code[2]
                 elif n == 4:
                     return post_code[4]
```

1.3 Data Overview and Additional Ideas

1.3.1 File Sizes

Data are written to csv files using data.py. And file sizes are as following:

```
new-york_new-york.osm   2.83GB
nodes.csv    1.03GB
nodes_tas.csv    30.1MB
ways_nodes.csv    349.7MB
ways_tags.csv    294.3MB
ways.csv    118.1MB
```

import files to sqlite

```
sqlite3
sqlite> .mode csv
sqlite> .import nodes_tags.csv nodes_tags
sqlite> .import nodes.csv nodes
sqlite> .import ways_nodes.csv ways_nodes
```

```
sqlite> .import ways_tags.csv ways_tags
sqlite> .import ways.csv ways
sqlite> .schema ways
```

1.3.2 Number of nodes

```
sqlite> SELECT COUNT(*) FROM nodes;
11561823
```

1.3.3 Number of ways

```
sqlite> SELECT COUNT(*) FROM ways;
1813535
```

1.3.4 Number of unique users

```
sqlite> SELECT COUNT(DISTINCT(e.uid)) FROM
   ...> (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
5076
```

1.3.5 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;
Rub21_nycbuildings,4884691
ingalls_nycbuildings,935684
MySuffolkNY,626317
woodpeck_fixbot,612055
SuffolkNY,580420
minewman,493614
Northfork,413359
ediyes_nycbuildings,271811
lxbarth_nycbuildings,233519
smlevine,219874
```

1.4 Additional Ideas

1.4.1 Data Imprement Ideas

```
When I submit queries looking for amenities there is no convenience markets like shoprite and w

When I look into this data, the most common key for value shoprite is name which is not that re

Anticipated Issues:
1. When someone wants to do a research about coverage of markets/supermarkes of new york area a
```

Here comes my solution:
1. Standardize keys for certain category values, for example, a supermarkets could be assigned
2. When a user contributes to the data a similar has already existed in the data, the existing
2. Encourage users not only contribute to common ones but also to some which can easily be forg

If above issue could be solved, then the data would be much more sufficient to answer more vari

1.4.2 Additional Data Exploration

Number of concerned amentity cafe and market

```
sqlite> SELECT COUNT(DISTINCT(id)) FROM nodes_tags
   ...> WHERE key = 'amenity' AND value = 'cafe';
1108


sqlite> SELECT COUNT(DISTINCT(id)) FROM nodes_tags
   ...> WHERE UPPER(value) = 'SHOPRITE';
31
```

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;
bicycle_parking,5086
restaurant,3332
school,3201
place_of_worship,3007
cafe,1108
fast_food,1031
bench,704
bank,643
fire_station,559
bar,489
```

1.5 Conclusion

The data set for New York is really large and there are so many users contributed to this map c