OpenStreetMap Data Case Study with SQL

Author: Qingyu Li

Date: Sep 26 2017

Map Area

Philadelphia, United States

- The OpenStreetMap Project: https://www.openstreetmap.org/relation/188022 (https://www.openstreetmap.org/relation/188022)
- Data Extract: https://mapzen.com/data/metro-extracts/metro/philadelphia_pennsylvania/ (https://mapzen.com/data/metro-extracts/metro/philadelphia_pennsylvania/)

I have lived in Philadelphia for a while and I would like to learn more about this fun city.

Problems Encountered in Map and Data Cleaning

After downloading the dataset for Philadelphia, we select a sample of the datafile and look for areas that need additional cleaning:

Street Name:

- Inconsistency in street name: 'St' for 'Street', 'Pkwy' for 'Parkway'. We need to update these names to make the street name consistent across all data points
- Misspelling of street names: 'Lane' is spelled as 'Line', 'Street' spelled as 'Sstreets',etc.
- City and state name are included in the street name. For example, "Baltimore Pike, Springfield, PA".

Zip code:

During auditing, we spot the following issue:

- Zipcode followed by mail box: 19148-9996
- State before zipcod: PA 19132

Auditing street Names

```
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()
    return street_types
def update_name(name, mapping):

    # YOUR CODE HERE
    m=street_type_re.search(name)
    if m not in expected:
        if m.group() in mapping.keys():
            name = re.sub(m.group(), mapping[m.group()], name)
    name = re.split(",|\#|\-|\;",name)[0]
    return name
st_types = audit(OSMFILE)
```

Using the audit code above, we print out part of the results below:

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

Output:

```
White Horse => White Horse
Redstone Ridge => Redstone Ridge
1 Brookline BlvdHavertown, PA 19083(610) 446-1234 => 1 Brookline BlvdHavertown
Market Street; Pennsylvania Route 452 => Market Street
Hillcrest Heights => Hillcrest Heights
```

Here we can see that all the issue with street names are updated.

Auditing zip code

```
def is_postcode(elem):
    """check if elem is a postcode"""
    return (elem.attrib['k'] == "addr:postcode" or elem.attrib['k'] == "postal_co
de")

def audit_postcode(postcodes, postcode):
    """ Get a full list of entries about postcode """
```

```
    postcodes[postcode].add(postcode)
    return postcodes
def audit_post(OSMFILE):
    """ match above function conditions """
    osm_file=open(OSMFILE, 'r')
    postcodes = 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_postcode(tag):
                    postcodes = audit_postcode(postcodes, tag.attrib["v"])
    osm_file.close()
    pprint.pprint(dict(postcodes))
audit_post(OSMFILE)
```

Create SQL database from cleaned XML data

Overview of Database

File Name	Size
Philadelphia_pennsylvania.osm	737.1 MB
philadelphia.db	531.4 MB
nodes.csv	274.8 MB
nodes_tags.csv	20.2 MB
ways_nodes.csv	94.6 MB
ways.csv	20.6 MB
ways_tags.csv	54.1 MB

Learn More about Philadelphia

# #

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

Number of nodes: 3289070

```
SELECT COUNT(*) FROM ways
```

Number of ways: 343066

```
SELECT COUNT(DISTINCT(e.uid)) \
            FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e
```

Number of unique users: 2340

```
SELECT value, COUNT(*) as num \
            FROM nodes_tags \
            WHERE key="amenity" \
            GROUP BY value \
            ORDER BY num DESC \
            LIMIT 1
```

Common ammenities: (u'school', 1613)

```
SELECT nodes_tags.value, COUNT(*) as num \
            FROM nodes_tags \
                JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value="restaura
nt") i \
                ON nodes_tags.id=i.id \
            WHERE nodes_tags.key="cuisine" \
            GROUP BY nodes_tags.value \
            ORDER BY num DESC\
            LIMIT 1
```

Popular cuisines: (u'pizza', 119)

We can see that the cuisine people like the most in Philadelphia is pizza.

Conclusion

The OpenStreetMap data of Philadelphia is of good quality. We found some typos in street spelling and some address contain city and state names as well as inconsistency in abbreviations in the street type. We spent a significant amount of time checking different fields and cleaning the dataset. However, there are still a lot of extra work needed to improve the data quality of this extract.

Suggestions for standardizing street names and control typos:

- We can work on some preset rules for data imput. Preset the common names selection, such as "Street", "Place", "Lane", "Pike", "Parkway", etc.
- We can develop a function to automatically clean the data periodically. This may be tedious at first, but with all data imput well controlled and database well maintained, the database will be easier to use and users do not need to spend a lot of time working on data cleaning.

Files

- Philadelphia_pennsylvania.osm: data downloaded from the OpenStreetMap project
- audit.py: audit and update street names

- data.py: parse and shape the osm data and create CSV files from OSM dataset
- dbwrite.py: read in the CSV files and create sql database
- mapparser.py: find unique tags in the data
- query.py: data exploration using sqlite