OpenStreetMap Data Wrangling Project

Map area

Cleveland, OH, United States

OpenStreetMap link

OSM extracted metro

Cleveland, OH is my hometown. I'm interested in exploring the OpenStreetMap data for Cleveland so I can identify problems with the map and improve the data for future use.

Problems with the map area

Following initial queries in the Cleveland OSM database, I identified 4 areas to focus on for data cleaning in this project: street names, postal codes, city names, and amenities.

Street names – I modified the code from the OpenStreetMap Data case study to standardize the street names in the Cleveland OSM data. In the street_names.py file, the xml file is parsed, the tags are identified as a street name, matched against a list of expected street types, and converted to the correct type if they do not match the list in this function:

```
def update_name(original_street_name, mapping=MAPPING):
    split_name = original_street_name.split(" ")
    abbr_street = split_name[-1]
    name = original_street_name
    if abbr_street in mapping:
        name = " ".join(split_name[:-1] + [mapping[abbr_street]])
    return name
```

Postal codes – Before data cleaning, some of the postal codes in the data set included "plus-four codes". One postal code tag was "Ohio", not a valid postal code. In the postcodes.py file, the xml file is parsed, the tags are identified as a postal

code, the postal codes are determined to be a valid postal code, and truncated to a 5-digit code ("plus-four codes" are removed). In these functions, the post codes are determined to be valid and truncated to 5-digit codes:

```
def is_valid_postcode(postcode):
    if postcode != "Ohio":
        return True

    return False

def truncate_postcode(postcode):
    if len(postcode) > 5:
        new_code = postcode[:5]
        return new_code

    return postcode
```

I decided to remove the "plus-four codes" because it was easier to standardize all the codes to a 5-digit code instead of adding the correct "plus-four code" for each node.

City – The Cleveland OSM extracted map is not limited to Cleveland city limits; other cities and towns in the surrounding metropolitan area are also included in the map. In the city.py file, the xml file is parsed, the tags are identified as cities, matched against a list of expected cities, and tags that are not a valid city are removed:

```
def update_city_list(cities):
    city_list = set()
    for city in cities:
        if is_valid_city(city):
            city_list.add(city)
    return city_list
```

Amenities – A query for the amenities tag indicated that most of the amenities were all lower-case and with underscores between words instead of spaces. I

decided to standardize the amenities to all have that format. These changes are made in the amenities.py file:

```
def update_amenities(amenities):
    new_amenities = set()
    for amenity in amenities:
        new_amenities.add(normalize_amenity(amenity))
    return new_amenities


def normalize_amenity(amenity):
    result = amenity.lower()
    result = result.replace(" ", "_")
    return result
```

All of these changes are incorporated into the data in the data.py file.

Overview Statistics

The cleveland.osm file is 276 MB.

Number of users

There are 554 distinct users in the data set:

```
sqlite> SELECT COUNT (DISTINCT(nodes_ways.uid))
   ...> FROM (SELECT uid FROM nodes UNION SELECT uid FROM ways) AS
nodes_ways;
554
```

Number of nodes

There are 1,164,752 nodes in the data set:

```
sqlite> SELECT COUNT(*)
```

```
   ...> FROM nodes;
1164752
```

Number of ways

There are 114548 ways in the data set:

```
sqlite> SELECT COUNT(*)
   ...> FROM ways;
114548
```

Top 10 amenities

```
sqlite> SELECT value, COUNT(*) as count
   ...> FROM nodes_tags
   ...> WHERE key='amenity'
   ...> GROUP BY value
   ...> ORDER BY count DESC
   ...> LIMIT 10
   ...> ;
place_of_worship|1896
school|1524
grave_yard|814
post_office|345
restaurant|311
parking|255
fast_food|184
fuel|135
library|117
```

hospital|91

Additional Ideas

Nodes marking ways vs. nodes marking locations

Of the 1,164,752 nodes in the data set, 1,152,841 describe ways:

```
sqlite> SELECT COUNT (DISTINCT(id))
   ...> FROM (SELECT * FROM nodes
   ...> JOIN ways_nodes
   ...> ON nodes.id=ways_nodes.node_id);
1152841
```

Only 1,571 nodes have a tag describing the address of the location:

```
sqlite> SELECT COUNT(*)
   ...> FROM nodes_tags
   ...> WHERE type='addr'
   ...> ;
1571
```

And only 6,673 nodes have a tag describing an amenity of the location:

```
sqlite> SELECT COUNT(*)
   ...> FROM nodes_tags
   ...> WHERE key='amenity';
6673
```

This data set would be more useful for determining the exact locations of streets and highways in the greater Cleveland area than it would be for learning about the address or use of specific locations.

Conclusions

When it comes to address and amenity data for the greater Cleveland area, the OpenStreetMap data has been programmatically cleaned and standardized in the course of this project, but it is not complete. A vast majority of nodes mark ways in the dataset – streets and highways. Therefore, I would suggest that this dataset would be useful for those interested in determining the exact locations of streets, highways, and other boundaries, rather than descriptive information about individual places.

Resources

Udacity sample project
OpenStreetMap documentation
Udacity project forums
Stackoverflow forums
SQL tutorials