Wrangle Open Street Map Data

Map Area

Poway, CA, United States of America

I chose this area because I am from a rural area nearby that did not contain enough data for the purposes of this project. This area includes the surrounding areas and cities.

Problems Encountered

Using audit_f.py I investigated several possible problem areas until I found the three that I considered the most relevant. These were:

- Incorrect or abbreviated street type
- Incorrect state for the addr:state tag
- Incorrect country for the is_in:country tag

Incorrect or abbreviated street type:

A total of 78 problems were found in addr:street and corrected using audit_f.py. These problems included errors such as wa for way, pe for peak, and py for Parkway which were found and corrected. Another problem was inconsistent names. In some entries Sr-78 was used and in others Highway 78 was used. Although it is correct that Highway 78 is a state route, local maps and tradition always use Highway 78 instead of Sr-78, so I decided to correct the Sr-78 entries and use Highway 78. The following code from audit_f.py was used in conjunction with other code to make the final changes to the erroneous entries.

```
if is_street_name(tag):
    m = street_type_re.search(tag.attrib['v'])
    name = tag.attrib['v']
    if m:
        street_type = m.group()
        street_type_set.add(street_type)
        if street_type  not in expected:
            if street_type in mapping.keys():
                old_street = tag.attrib['v']
                tag.attrib['v']  = re.sub(street_type_re, mapping[street_type], name)
                new_street = tag.attrib['v']
                street_changes[old_street].append(new_street)
```

Incorrect state for the addr:state tag:

As I was searching the data for incorrect data I came across an unexpected, but common error. The state was incorrect. Most of the erroneous entries were variations of CA such as ca or Ca. Some of the entries were simply the state spelled out as California. Some unexpected results were AZ and Elisa Lane. The following code from audit_f.py was used in conjunction with other code to make the final changes to the erroneous entries.

```
if is_state(tag):
    state_name = tag.attrib['v']
    if state_name != 'CA':
       old_state = tag.attrib['v']
       tag.attrib['v']  = 'CA'
       new_state = tag.attrib['v']
       state_changes[old_state].append(new_state)
```

Incorrect country for the is_in:country tag:

Another unexpected problem in the data was the incorrect tag is_in:country. Using audit_f.py I was able to find 15 erroneous entries. These errors were due to some users using the abbreviation USA instead of spelling out United States of America. I decided to make the change for the purpose of uniformity rather than error due to the fact that 33,161 entries used the spelled-out form of country versus only 15 that used abbreviations. The following code from audit_f.py was used in conjunction with other code to make the final changes to the erroneous entries.

```
if is_country(tag):
    country_name = tag.attrib['v']
    if country_name != 'United States of America':
        old_country = tag.attrib['v']
        tag.attrib['v']  = 'United States of America'
        new_country = tag.attrib['v']
        country_changes[old_country].append(new_country)
```

Data Overview and Exploration

File Sizes:

PowayCA.osm	212 MB
Poway_DB.db	70 MB
nodes.csv	66 MB
nodes_tags.csv	41 MB
ways.csv	6 MB
ways_nodes.csv	18 MB
ways_tags.csv	10 MB

Number of Unique Users:

```
cur.execute("""SELECT COUNT(DISTINCT(u.uid))
                FROM (SELECT uid FROM Node UNION ALL SELECT uid FROM ways) u
            ;""")
pp.pprint(cur.fetchall())
```

The above code from sql_query.py yields 1357 unique users. The top 10 users were determined by the following code.

```
cur.execute("""SELECT e.user, COUNT(*) as num
                FROM (SELECT user FROM Node UNION ALL SELECT user FROM ways) e
                GROUP BY e.user
                ORDER BY num DESC
                LIMIT 10
            ;""")
pp.pprint(cur.fetchall())

[(u'n76', 98296),
 (u'woodpeck_fixbot', 70608),
 (u'SatGarcia', 63264),
 (u'evil saltine', 57829),
 (u'TheDutchMan13', 37442),
 (u'Adam Geitgey', 27903),
 (u'CSanders0', 26218),
 (u'mapman44', 18988),
 (u'riseagainstme', 15264),
 (u'NewsBees', 12994)]
```

Number of Nodes:

```
cur.execute("""SELECT COUNT(*)
               FROM Node
            ;""")
print('There are {} nodes.'.format(cur.fetchall()))
```

There are [(792123,)] nodes.

Number of ways:

```
cur.execute("""SELECT COUNT(*)
               FROM ways
            ;""")
print('There are {} ways.'.format(cur.fetchall()))

There are [(76410,)] ways.
```

The most common land use and the name of the vineyard from the results:

```
cur.execute("""SELECT value, COUNT(*) as num
                FROM nodes_tags
                WHERE key='landuse'
                GROUP BY value
                ORDER BY num DESC
                LIMIT 10
            ;""")
pp.pprint(cur.fetchall())

[(u'quarry', 7),
 (u'industrial', 6),
 (u'reservoir', 2),
 (u'vineyard', 1),
 (u'residential', 1),
 (u'commercial', 1)]

cur.execute("""SELECT nodes_tags.value
                FROM nodes_tags
                JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE key = 'landuse')lu
                ON nodes_tags.id = lu.id
                JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value = 'vineyard')v
                ON lu.id = v.id
                WHERE nodes_tags.key = 'name'
            ;""")
name = cur.fetchone()
for n in name:
    name = n
print('The name of the vineyard is {}.'.format(name))

The name of the vineyard is Orfilia Winery.
```

The 10 most common tourist attractions and the types of artwork:

```
cur.execute("""SELECT value, COUNT(*) as num
                FROM nodes_tags
                WHERE key='tourism'
                GROUP BY value
                ORDER BY num DESC
                LIMIT 10
            ;""")
print('The 10 most common tourist attractions are:')
pp.pprint(cur.fetchall())

The 10 most common tourist attractions are:
[(u'information', 53),
 (u'viewpoint', 39),
 (u'artwork', 32),
 (u'picnic_site', 26),
 (u'hotel', 15),
 (u'attraction', 11),
 (u'apartment', 3),
 (u'theme_park', 1),
 (u'museum', 1),
 (u'motel', 1)]



cur.execute("""SELECT nodes_tags.value, COUNT(*) as num
               FROM nodes_tags
               JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value = 'artwork')ar
               ON nodes_tags.id = ar.id
               WHERE nodes_tags.key = 'artwork_type'
               GROUP BY value
               ORDER BY num DESC
               LIMIT 10
        ;""")
print('The 10 most common artwork types:')
pp.pprint(cur.fetchall())

The most common artwork types:
[(u'sculpture', 21),
 (u'statue', 3),
 (u'mural', 2),
 (u'stone', 1),
 (u'graffiti', 1),
 (u'bust', 1),
 (u'architecture', 1)]
```

Conclusion and Additional Suggestions for Improvement

Conclusion:

The changes documented above address several key areas of quality data. Accuracy is addressed in the street audit by accurately representing the street types. Accuracy was also addressed in the state audit where several states were wrong or incorrectly typed. Consistency was addressed in the country audit where no incorrect data was found but several inconsistent entries were found.

Suggestions for Further Improvement:

During the above audit I encountered a glaring deficit in the street name audit. Poway's proximity to the border with Mexico, and the fact that it was a colony of Spain, both lead to a very large number of street names that are of Spanish origin. Instead of 7th Street the name would be Calle 7. A total of 1267 unique street types were found using audit_f.py. Below is a list of the 10 most common not in the expected street names. The following function is from the audit_f.py.

```
def most_common_street_type(street_list):
    new_street_list = []
    for st in street_list:
        if st not in expected:
            new_street_list.append(st)
    return sorted(set(new_street_list), key=street_list.count)[-10:]
```

```
['Row', 'Valle', 'Mar', 'Vista', 'Real', 'Cove', 'Point', 'Glen', 'Terrace', 'Circle']
```

The above results clearly demonstrate the difficulty of cleaning data that is in multiple languages. Four of the top ten street types not in expected are in Spanish. Another area for improvement is apparent in the above query of tourist attractions where "apartment" is listed three times in error.

Resources

- https://docs.python.org/2/library/xml.etree.elementtree.html
- https://towardsdatascience.com/processing-xml-in-python-elementtree-c8992941efd2?gi=7d382da93808
- https://stackoverflow.com
- https://www.w3schools.com
- Examples and starter code from the several portions of the course