Wrangle OpenStreetmap Data for Mesa/Gilbert, AZ

Author: Andrew Marksberry

Map Area

Mesa/Gilbert, AZ

- https://www.openstreetmap.org/export#map=12/33.3282/-111.6589

This map is of my current home and I have lived here for 2 years and feel like I know the area well. I am interested to see the data that is contained in OpenStreetMap.org and am hoping to assist with making the data better.

Problems Encountered in the Map

After downloading the area data, I ran some functions in audit.py to find problems with the data and correct the issues.

- Abbreviated streets types
- Incorrect postal codes
- Incorrectly formatted phone numbers

Abbreviated streets types

While auditing the data I noticed there were a few street types that were abbreviated. (ex. Ave, Rd, St) I then ran an update function to spell out the street types, (ex. Avenue, Road, Street).

```
#Perform the update of each street that needs fixed
def update(name, mapping):
    word = name.split()
    for w in range(len(word)):
        if word[w] in mapping:
            word[w] = mapping[word[w]]
    name = ' '.join(word)
    return name
```

This updated the names such as "E Ray Rd" to "E Ray Road".

Postal Codes

Most postal codes were listed as 5-digit numbers, however, there were a few that had the delivery area 4 digits attached so using the below code I cleaned the postal codes to all be 5 digits.

```
def update_postcode(postcode):
    match = re.match(r'^\D*(\d{5}).*', postcode)
    clean_postcode = match.group(1)
    return clean_postcode
```

Incorrect formatted phone numbers

The phone number format used by OpenStreetMaps is either uses spaces or '-' between the digits of the phone number and contains the country code in the beginning. Several records contained parenthesis, some contained no country code, etc. Using the formula below I cleaned the phone numbers to the appropriate format.

```
def updatephone(phone_number, mapping):
    phone_number = phone_number.replace('+','').replace('-', '').replace('(','').replace(')','').replace(' ','').replace('.','')
    if 'x' in phone_number:
        phone_number = phone_number[:10]
    elif phone_number[0] == '1':
        phone_number = phone_number[1:]
    else:
        phone_number
    phone_number = '+1-'+phone_number[0:3]+'-'+phone_number[3:6]+ '-'+phone_number[6:]
    return phone_number
```

Data Overview

With the clean data I imported the information into SQLite and used queried the data using python SQLite3. All codes can be found in the “SQL Data.py” file

File Sizes

Mesamap.osm	59,997KB
Mesamap_nodes.csv	21,238KB
Mesamap_nodes_tags.csv	720KB
Mesamap_ways.csv	2,380KB
Mesamap_ways_nodes.csv	7,407KB
Mesamap_ways_tags.csv	5,591KB

Number of Nodes

```
cur=conn.cursor()
cur.execute("SELECT count(id) FROM nodes")
data = cur.fetchone()
print ("Number of nodes:",data[0])
```

```
1080304
```

Number of Ways

```
cur=conn.cursor()
cur.execute("SELECT count(id) FROM ways")
data = cur.fetchone()
print ("Number of ways:",data[0])
```

```
158552
```

Number of Unique Users

```
cur=conn.cursor()
cur.execute("SELECT DISTINCT count(*) FROM (SELECT uid FROM nodes UNION SELECT uid FROM ways)")
data = cur.fetchone()
print ("Number of unique users:",data[0])
```

```
1055
```

Top 10 Contributors

```
cur=conn.cursor()
cur.execute("""SELECT data.user, COUNT(*) as number
            FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) as data
            GROUP BY data.user
            ORDER BY number DESC
            LIMIT 10;""")

data = cur.fetchall()

print ("Top 10 contributing users")
for row in data:
    print (row)
```

```
('AJ Riley', 139900)
('HJUdall', 133224)
('TheDutchMan13', 128260)
('MisterPhilip', 117214)
('Dr Kludge', 97482)
('chachafish', 73704)
('adenium', 66664)
('David Maciaszek', 52636)
('Gkieta', 50108)
```

```
('ParagonPrime', 29780)
```

Top 10 Amenities (Nodes and Ways)

```
cur=conn.cursor()
cur.execute("""SELECT data.value, COUNT(*)
            FROM (SELECT value, key FROM nodes_tags UNION ALL SELECT value, key FROM ways_tags) as data
            WHERE data.key='amenity' and data.value<>''
            GROUP BY data.value
            ORDER BY Count(*) DESC
            LIMIT 10;""")

data = cur.fetchall()

print("Top 10 amenities")
for row in data:
    print (row)
```

```
('waste_disposal', 662)
('parking', 572)
('fast_food', 228)
('restaurant', 194)
('post_box', 146)
('bench', 132)
('school', 126)
('clinic', 122)
('place_of_worship', 106)
('shelter', 106)
```

Conclusion

After scrubbing the data, it was hard to find huge problems with the data, there were a few errors that I corrected but the data seems to be added pretty accurately or corrected often. I feel like OpenStreetMaps could improve the data they receive by setting up data validation on fields. They may run into issues though since different countries may have different values and it could ultimately stop users from entering the correct data because they have to conform to the validation rules.