Erica Racine

Udacity DAND: Data Wrangling, OpenStreetMap Project

Map Area: **Lowell, MA, United States**

I chose Lowell because I grew up not too far from the city. Recently someone had told me they had gone there for vacation. I hadn't been there in years and was curious about how the city has changed.

https://www.openstreetmap.org/relation/1805416#map=13/42.6365/-71.3269

http://overpass-api.de/query_form.html

Problems Encountered in the Map

After doing some testing (lowellplayground.py), I found several types of recurring errors. Three of the most common were:

1. Street name abbreviations (for example, Graniteville Rd, Richardson Rd, Chelmsford St.)
2. Differing phone number formats (for example, +1 978 937 7667, (978) 454-9332, 978-459-6637)
3. Certain zip codes that are out of area (for example, 01879) Lowell zip codes: 01850-01854

Street name abbreviations

To fix the issue of street name abbreviations, I first audited the "addr:street" values and compiled a list of problematic street types. I then cleaned the data by iterating over each "addr:street" value and if the street type was found to be problematic (not in the list of "expected" street names), I updated the street name according to the relevant mapping.

Code used for updating street name:

```
# If the match found is located in the mapping, update the match
def update_name(name, mapping):
   m = street_type_re.search(name)
   if m.group() in mapping.keys():
      name = re.sub(m.group(), mapping[m.group()], name)
```

Problematic Phone Numbers:

Printing out a list of phone numbers from the Lowell, MA dataset showed that the numbers were in complete disarray.

Small sample of phone numbers:

```
+1-978-251-4050
+1-978-649-4400
(978) 453-1112
+1-978-251-5155
978-275-9585
978-459-6637
```

Although preferred format is certainly subjective, after doing a little research it seemed to me that xxx-xxx-xxxx was the generally accepted preferred format for phone numbers. To clean the phone numbers, I pulled a list of the problematic phone numbers and reformatted them using this function:

```
def update_phone(number):

    if any(x in number for x in phone_problems):

        number = number.lstrip("'+1'").replace('(', '').replace(')', '')
        if number[0]==" " or number[0]=="-":
            number=number.lstrip("'', '-'")
            number=number.lstrip("-")
        number=number.replace(" ", "-")
        if number[7] != "-":
            number = number[:7] + '-' + number[7:]

    return number
```

Postal Codes

I performed this query:

```
SELECT value, count(*)
FROM ways_tags
WHERE key='addr:postcode'
GROUP BY value
ORDER BY count(*) DESC;
```

to see a list of the postal codes and their frequency. These are the results:

```
01852|325
01876|85
01824|70
01863|41
```

```
01854|27
01851|23
01879|16
01826|8
01886|2
01284|1
01810|1
01825|1
01850|1
01862|1
08163|1
```

After seeing that a majority of the zip codes begin with “018”, I researched and confirmed that each of the “018” zip codes listed is in the Lowell, MA area, so actually these are not a problem afterall. Further filtering:

```
SELECT value, count(*) as num
FROM ways_tags
WHERE key='addr:postcode'
AND (value NOT LIKE '018%')
GROUP BY value;
```

Revealed that there are only two zip codes that do not begin with “018”:

```
01284|1
08163|1
```

The 08163 seemed like it could be a typo. Further querying:

```
#To find “id”
SELECT * FROM ways_tags WHERE key = 'addr:postcode' and value = '08163';

“To find “street”
SELECT value FROM ways_tags WHERE key = 'addr:street' and id=212374466;
```

Revealed that the zip code ‘08163’ is linked to Groton Road. I looked up and found a Groton Road in ‘01863’, furthering the likelihood that this postal code problem was due to a typo.

A similar query:

```
SELECT * FROM ways_tags WHERE key = 'addr:postcode' and value = '01284';

SELECT value FROM ways_tags WHERE key = 'addr:street' and id=212363965;
```

revealed that ‘01284” was linked to “Colonial Road”. A google search revealed that there is a Colonial Road in the common Lowell area zip code “01824”, so this being another typo is a possibility.

Overview of the Data

This section will include key statistics about the dataset and files, some notable findings and the queries used to gather this information.

File Sizes

lowell.xml	96.7 MB
lowellsql1.db	57.9 MB
nodes.csv	37.4 MB
ways_nodes	11.4 MB
nodes_tags.csv	4.9 MB
ways.csv	3.7 MB
ways_tags	3.7 MB

Number of Nodes

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

```
407876
```

Number of Ways:

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

```
55039
```

Number of Unique Users:

```
SELECT COUNT(DISTINCT(uid))
FROM
 (SELECT uid FROM nodes
 UNION ALL
   SELECT uid FROM ways);
```

```
222
```

Additional Explorations

Most popular types of shops:

```
SELECT value, count(*) as num
FROM nodes_tags
```

```
WHERE key='shop'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

```
hairdresser|8
alcohol|5
supermarket|5
clothes|4
convenience|3
beauty|2
car_repair|2
doityourself|2
gift|2
greengrocer|2
```

Most popular cuisines:

```
SELECT value, count(*) as num
FROM nodes_tags
WHERE key='cuisine'
GROUP BY value
ORDER BY num DESC;
```

```
pizza|7
chinese|4
coffee_shop|3
donut|2
french|1
greek|1
italian|1
italian_pizza|1
mexican|1
pizza_&_subs|1
sandwich|1
seafood|1
thai|1
vietnamese|1
```

When were the first 10 restaurants added to the dataset:

```
SELECT nodes.timestamp, nodes_tags.value
FROM nodes
JOIN nodes_tags
ON nodes.id=nodes_tags.id
WHERE nodes_tags.key='amenity' AND nodes_tags.value='restaurant'
```

```
ORDER BY timestamp
LIMIT 10;
```

```
2010-12-04T20:32:44Z|restaurant
2010-12-04T20:32:44Z|restaurant
2010-12-04T20:32:44Z|restaurant
2011-11-27T14:26:25Z|restaurant
2012-05-06T22:43:55Z|restaurant
2012-09-09T00:20:11Z|restaurant
2012-09-11T17:40:36Z|restaurant
2012-09-11T17:47:09Z|restaurant
2013-01-16T18:44:05Z|restaurant
2014-03-09T00:21:43Z|restaurant
```

When was the first pizza place added to the dataset:

```
SELECT nodes.timestamp, nodes_tags.value
FROM nodes
JOIN nodes_tags
ON nodes.id=nodes_tags.id
WHERE nodes_tags.key='cuisine' AND nodes_tags.value='pizza'
ORDER BY timestamp
LIMIT 1;
```

```
2012-09-11T17:40:36Z|pizza
```

Additional ideas

What initially drove me to select Lowell, MA, as the area that I would like to investigate with OpenStreetMap was that after hearing that a friend was going to visit the city on vacation, I was curious as to how the city had changed since the last time I was there, which was at least 10 years ago. I was excited to read in the OpenStreetMap wiki that “Deleted nodes remain in the database forever with unchanged id, but with visible=false instead of visible=true.” Unfortunately, a simple query revealed that there were no such “visible” attributes in my dataset.

```
SELECT value, count(*)
FROM nodes_tags
 WHERE key='visible';
```

```
|0
```

The documentation (https://wiki.openstreetmap.org/wiki/Find_the_id_of_a_deleted_node) listed other ways of potentially finding the id of a deleted node (including WhoDidIt and OSMCHA) , but after exploring several I found that the majority of them were most effective in exploring recent changes to an area, or required a whole lot of memory. Considering that the data in the Lowell, MA, dataset goes back many years, the oldest being:

```
SELECT timestamp
FROM nodes
ORDER BY timestamp
LIMIT 1;
```

```
2007-10-11T17:53:20Z
```

I think that finding a way to more easily access historical data would be a worthwhile pursuit. Certainly factors like memory with large datasets will be a consideration, but as OpenStreetMap evolves in time, being able to retrieve data on how areas have changed seems to be a very important feature.

Conclusion

Although there are many great insights to be gleaned from this dataset, it is clear that there is still a lot of untapped value. I believe that consistency of the data after the cleaning of the street name types and phone number formats will allow for easier querying in the future. I would also like to see more possibilities for historical data.