Wrangling Data

Anthony Cassetta

Problems encountered in map

- Dirty data encounter in the “tag” tag’s with attribute “addr:street”

The data for street types (street, avenue, parkway…etc.) are inconsistent (see figure 2) in their abbreviation. This could lead to ambiguous or inconsistent statistics. So the data was audited using a regular expression to identify problem street types. The audit results were used to build a dictionary of keys containing corrected values.

```
------------Street------------
#501: 1
1702: 1
6: 1
Ave: 73
Ave.: 14
Avenue: 294
Boulevard: 6
Boylston: 2
Broadway: 42
Cambrdige: 1
Center: 8
Circle: 1
Court: 5
Ct: 2
Dr: 1
Drive: 31
Ext: 1
Fenway: 2
floor: 1
Greenway: 2
Hall: 1
Hampshire: 1
Highway: 2
```

Figure 1 Sample of audit output

- Dirty data encounter in the “tag” tag’s with attribute “addr:state”

The data present in the address state field was in consistent. (see figure 2). Some data points were the USE state code for Massachusetts, MA, others the full name and yet more a combination of both.

```
---------------State------------
MA: 882
ma: 1
Ma: 2
MA- MASSACHUSETTS: 32
Massachusetts: 6
MASSACHUSETTS: 1
```

Figure 2 Sample of audit output

- Dirty data encounter in the “tag” tags with attribute “addr:postcode”

The postal code data contained inconsistencies it varies between the 5-digit standard code and the new 9-digit standard code. I identified this error within the XML file and as it’s occurrences were low chose not to use it for demonstration purposes.

Data cleaned programmatically

As I said above, to clean the data within the street attribute a map was build based off of the audit report. The method pictured in figure 3 takes a name compares its street type to the keys of the map and if a match is found, updates the street type. Example “St” becomes “Street”.

```
1  '''Takes a given street name and compares its street type against the dictoary keys.
2  If an update is needed an update is made, else none is made.'''
3  def update_name(name, mapping):
4      new_name = ""
5      m = street_type_re.search(name)
6      if m:
7          street_type = m.group()
8          #print(street_type)
9          if street_type in mapping.keys():
10             new_name = re.sub(street_type_re, mapping[street_type], name, count=0, flags=0)
11             print('updated ' + name, 'to ' + new_name)
12             name = new_name
13         #print(new_name)
14     return name
15
```

Figure 3 Update Name method

Following a similar but more straight forward methodology. After the audit reported that all state fields were in fact some value representing Massachusetts. I decided to indiscriminately update all state fields to match the standard US state code, MA.

```
16
17 '''Ups the state code values.'''
18 def update_state(state):
19     state = 'MA'
20     return state
21
```

Figure 4 update State method

Overview of the data

Taken from openstreetmap.org.

Location, Cambridge Massachusetts USA

https://www.openstreetmap.org/export#map=14/42.3747/-71.1127

Uncompressed map data file size: 77.9 MB

XML Tag counts

- 'member': 27004
- 'nd': 429201
- 'node': 348111
- 'osm': 1
- 'relation': 404
- 'tag': 164293
- 'way': 56539

SQL file sizes

SQLite database "Cambridge.db":	54.8 MB
Nodes.csv:	26.9 MB
Nodes_tags.csv:	1.83 MB
Ways.csv:	3.54 MB
Ways_nodes:	9.27 MB
Ways_tags:	4.27 MB

number of unique users:

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

Result: 878

```
sqlite> SELECT COUNT(DISTINCT(e.uid))
   ...> FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
878
sqlite>
```

number of nodes and ways

SELECT COUNT(*) FROM nodes;

Result: 348111

```
sqlite> SELECT COUNT(*) from nodes;
348111
sqlite>
```

SELECT COUNT(*) FROM ways;

Result: 56539

```
sqlite> SELECT COUNT(*) from ways;
56539
sqlite>
```

Number of distinct streets

SELECT COUNT(DISTINCT(e.key))
FROM (SELECT key FROM nodes_tags UNION ALL SELECT key from ways_tags WHERE key = 'Street') e;

Result: 300

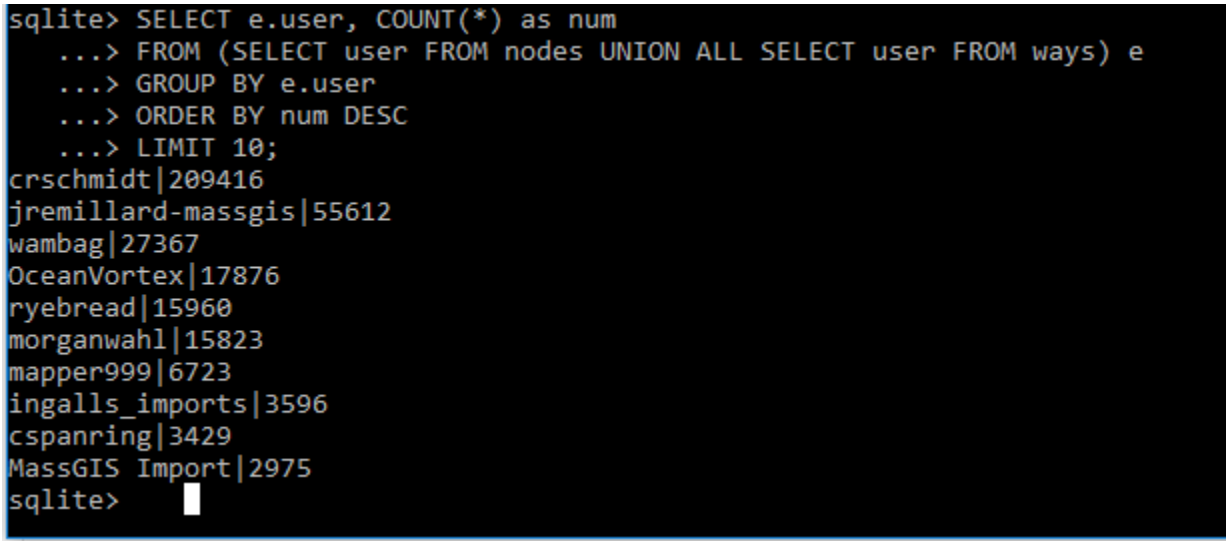
```
sqlite> SELECT COUNT(DISTINCT(e.key))
   ...> FROM (SELECT key FROM nodes_tags UNION ALL SELECT key from ways_tags WHERE key = 'Street')e;
300
sqlite>
```

Other ideas about the dataset

Top ten contributing users

```
SELECT e.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
GROUP BY e.user
ORDER BY num DESC
LIMIT 10;
```

Crschmidt	209416
jremillard-massgis	55612
wambag	27367
OceanVortex	17876
ryebread	15960
morganwahl	15823
mapper999	6723
ingalls_imports	3596
cspanring	3429
MassGIS Import	2975



Number of users with only one post

```
SELECT COUNT(*)
FROM
        (SELECT e.user, COUNT(*) as num
         FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
         GROUP BY e.user
         HAVING num-1) u;
```

Result: 250

```
sqlite> SELECT COUNT(*)
   ...> FROM
   ...>     (SELECT e.user, COUNT(*) as num
   ...>      FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
   ...>      GROUP BY e.user
   ...>      HAVING num=1)  u;
250
sqlite>
```

Total posts

SELECT COUNT(*) as num FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e;

Result: 404650

```
sqlite> SELECT COUNT(*) as num from (SELECT user FROM nodes UNION ALL SELECT user FROM ways)e;
404650
sqlite>
```

Conclusion

I find it interesting that the user with the highest number of posts has more than three times that of the user with the second most posts. The highest poster in our sample contributed 209,416 posts. Out of our total count of 404,650. Meaning our top user contributed ~52% of all posts in our sample set of data from Cambridge, MA.

The top ten contributed 358,777 posts cumulatively. Meaning the top ten contributed ~90% of the total posts. Through this exploration we have also discovered 878 unique user ID's of those only 10 users contribute ~90% of all posts related to our sample. That's ~90% of posts coming from ~1% of users.

We also know that 250 users have only one post contributed. Meaning that of our sample population 28.5% of our users contribute to the Open Street Map only once.

Potential Solution

From a business perspective, Open Street Map may want to consider some form of reward system for users. Both to incentivize more users to contribute and to reward top performers such as our top ten shown above. Specifically, a system of achievements, like in a video game.

Benefits

1. Using an achievement structure as a positive way to encourage engagement and mark progress. With little financial impact to the organization.
2. Studies have shown gamification of learning and working can have a strong positive impact

I**ssues**

1. Implementing an achievement system would take some time and intelligent design initially.
2. It is possible users will create frivolous updates in order to farm points
3. More users entering posts about a specific area increases the likely hood of dirty data. Humans tend to make mistakes.

References:

Data Wrangling SQL Schema

https://gist.github.com/swwelch/f1144229848b407e0a5d13fcb7fbbd6f

Example SQL project

https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md

Opensteetmaps.org sample area of Cambridge MA

Location, Cambridge Massachusetts USA

https://www.openstreetmap.org/export#map=14/42.3747/-71.1127

Gameification of learning and instruction

Game based methods of instruction