openstreetmap

December 28, 2017

1 Wrangle OpenStreetMap Data

This project wrangles the geographical data of the Boston, MA, USA area to facilitate its analysis and allow the process of storage, querying, and aggregation of this information using SQL. Data is loaded from the OpenStreetMap (OSM) platform, which is a collaborative project to create a free editable map of the world. The first section describes the problems found at the time of the data cleaning process and the methods used to resolve them. The next section includes general statistics of the dataset computed, such as file sizes, number of nodes and ways, unique users and type of nodes, like cafes, shops etc. The last part includes suggestions to improve the analysis with a discussion about the benefits as well as some anticipated problems of their implementation.

1.0.1 Map Area

I chose Boston because I lived there for almost one year studing English. I would like to take advantage of this project to learn more about the city's geography. The data was loaded from: * https://www.openstreetmap.org/relation/2315704#map=11/42.3195/-70.9473

2 Problems encountered in the map

After auditing the dataset, the main problems founded were: * **Over abbreviated street names.** Example:

```
- Townsend St
- Townsend St.
- Townsend Street
```

- **Address format:** Addresses could be in two different formats:

- All the address information put together. Example: `xml <tag k="address" v="9 Townsend Street, Boston, MA"/>`
- Information divided into segments. Example: `xml <tag k="addr:city " v="Boston"/>`

```
<tag k="addr:street " v="Townsend Street"/>

<tag k="addr:state" v="MA"/>
```

```
<tag k="addr:housenumber" v="9"/>
```

- **Inconsistent postcodes:** Postcodes were in three different formats. Example: “‘
- MA 02138
- 02138-2706
- 02138 “‘ In the case that the key is ’address’, beside having those formats, some of them don’t have postcode or have the value ’00’.
- **Other postcodes:** The data includes information about some surrounding cities.

Below there’s an explanation of some of these issues:

2.0.1 Overabbreviated street names

To handle overabbreviated names, a map with all the different types of abbreviations and their complete names was created. Using the following function, which takes the addresses and the map described as a input, addresses can then be modified into a single format.

```
def update_name(name, mapping):
    m = street_type_re.search(name)
    if m:
        street_type = m.group()
        if street_type in mapping.keys():
            name = re.sub(m.group(), mapping[m.group()], name)

    return name
```

2.0.2 Inconsistent postcodes

Three different format of postcodes were founded. A function was created to transform all the postcodes to the format ’02138’

```
def update_postcode(postcode):
    if '-' in postcode:
        pos = postcode.find('-')
        return postcode[:pos]
    elif 'MA' in postcode:
        return postcode[3:]
    else:
        return postcode
```

2.0.3 Other postcodes

To audit the postcodes, a dictionary was built with the information in the sample dataset that contains all the zip codes and the time that they appear in the dataset.

```
def audit_postcodes(file_in):
    post = {}
    for element in get_element(file_in, tags=('node', 'way')):
        for child in element:
            if child.tag == 'tag':
                key = child.attrib['k']
                if 'address' in key:
                    v = child.attrib['v']
                    addr_list= v.split(" ")
                    value = addr_list[-1]
                    if value == 'MA' or value == '00':
                        continue
                elif 'postcode' in key:
                    value = child.attrib['v']
                else:
                    continue

                if value in post.keys():
                    post[value]+=1
                else:
                    post[value]=1

    return post
```

The result of that function was:

```
{'02186': 1, '02215': 5, '02124': 2, '02126': 2, '02127': 3, '02144': 2, '02109': 2, '02142': 4
```

Slighty more than half of the postcodes above don't belong to Boston but all of them are in Massachussets, Boston's state. It seems some adjacent cities were included in the dataset. To check it the following query shows the cities available in the dataset.

```
SELECT tags.value as City, count(*) as cant
FROM (select * from nodes_tags UNION ALL select * from ways_tags) as tags
WHERE tags.key = 'city' group by tags.value
ORDER BY count(*) DESC
LIMIT 15;")
```

City	Cant
Boston	1160
Cambridge	596
Malden	413
Arlington	311
Somerville	253
Jamaica Plain	108
Chelsea	67
Brookline	58

City	Cant
Quincy	55
Medford	47
Weymouth	34
Brighton	29
Chestnut Hill	23
Watertown	23
Allston	22

As it was predicted, the dataset includes information about nearby cities, where only 34.2% correspond to Boston.

3 Overview of the data

This section shows statistics that describe the dataset. In order to obtain them the cleaned data was imported into SQL, and statistics were extracted using queries.

3.0.1 Size of the file

```
boston_ma.osm ......... 428.5 MB
sample.osm ............ 9 MB
StreetMap.db .......... 322.3 MB
nodes.csv ............. 157.2 MB
nodes_tags.csv ........ 17.1 MB
ways.csv .............. 20.5 MB
ways_tags.csv ......... 22.2 MB
ways_nodes.cv ......... 54.3 MB
```

3.0.2 Number of unique users

```
SELECT COUNT(DISTINCT(users.uid)) AS NumUsers
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) AS users;

>>1,414 uniques users
```

3.0.3 Number of nodes

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

>>1,952,745 nodes
```

3.0.4 Number of ways

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

>>311,038 ways
```

3.0.5 Top 10 contributing users

```
SELECT users.user, count(*) as total, round(count(*)/2263783.0*100,2) as percentage
FROM (SELECT user FROM ways UNION ALL SELECT user from nodes) as users
GROUP BY users.user
ORDER BY total DESC
LIMIT 10;
```

user	total	percentage(%)
crschmidt	1195075	52.79
jremillard-massgis	428673	18.94
wambag	110955	4.9
OceanVortex	90998	4.02
morganwahl	67051	2.96
ryebread	65966	2.91
MassGIS Import	58640	2.59
ingalls_imports	32453	1.43
Ahlzen	28321	1.25
mapper999	14697	0.65

3.1 Additional information

3.1.1 Number of the 15 top amenities

```
SELECT  value, count(*) as cant
FROM nodes_tags
where key = 'amenity'
group by value
order by cant desc
limit 15;
```

Amenity	Cant
bench	1078
restaurant	708
school	93
bicycle_parking	326
cafe	279
place_of_worship	277
library	272
fast_food	207
bicycle_rental	138
post_box	125
waste_basket	114
parking	97
fire_station	90
bank	87

Amenity	Cant
bar	73

3.1.2 Top religions

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

value	num
christian	247
jewish	9
unitarian_universalist	2
buddhist	1
muslim	1

3.1.3 Top 10 cuisines

```
SELECT value , count(*) as num
FROM (SELECT distinct id FROM nodes_tags WHERE value = 'restaurant' ) as ids JOIN nodes_tags on
WHERE key = 'cuisine'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

value	num
pizza	44
american	39
chinese	32
italian	32
mexican	32
indian	22
thai	19
asian	15
japanese	14
regional	12

4 Other ideas about the dataset

4.0.1 Addresses' format standardization

As we discussed previously, addresses can be found in two formats. The following queries show the total number of addresses for each format that can be found in the database.

- **Format 1: Information divided into segments**

```
SELECT COUNT(ids.id)
FROM (SELECT DISTINCT id
        FROM nodes_tags
        WHERE type = 'addr'
        UNION ALL
        SELECT DISTINCT id
        FROM ways_tags
        WHERE type  = 'addr') as ids;
```

```
>> 8772
```

- **Format 2: All the address information put together**

```
SELECT COUNT(ids.id)
FROM (SELECT DISTINCT id
        FROM nodes_tags
        WHERE key = 'address'
        UNION ALL
        SELECT DISTINCT id
        FROM ways_tags
        WHERE key  = 'address') as ids;
```

```
>> 558
```

Having different format difficults the process of obtaining information as the data is not unified in a single format. To avoid this problem, I propose that users must fill predeterminate fields like "City", "Postal Code", "State", "Street", "House number". By doing so addresses can be ordered by each of the fields, facilitating queries and other uses of the data.

Another consideration is to put a roll list in the field "Street" with the most common types of addresses like road, street, boulevard, avenue, etc. as a safeguard against over abbreviated street names.

Making a more user-friendly interface can have the advantage of unifying the information into a single format, but it does not necessarily make it the correct way of retrieving data. It is not possible to ensure that each user will fill in the fields with the required information. On the other hand, making the user fill in more fields makes the process of entering the data slower and more tedious. Moreover, the user does not always have the complete information to fill in all the fields, a situation that may discourage participation.

5 Conclusion

The quality of the data wasn't bad, although it had a lot of typing errors. By doing the correct cleansing it was possible to browse through the information.

83.61% of the data was submitted by only 5 users, 52.79% was submitted by the user that contributed the most. . This means that the information is obtained from a very limited pool of sources, which can make it susceptible to biases.

An easier-to-use interface could help to avoid some of these problems while also motivating people to participate. For example, giving points to the most active users could result in a more interactive process that increases the amount of data available and the diversity of sources.