Data Science for Business Nanodegree - P3
Cleaning Data from OpenStreetMap

Fernando Maciel Motta

Map Area

Boston, MA, United States

- https://www.openstreetmap.org/relation/2315704

I chose this region because Boston has always appealed to me as a place where I'd like to live in the future, and I figured querying its database might give me more insight into it.

Problems Encountered in the Map

The main problems encountered while auditing the data were related to the street names. The problems were:

- Inconsistently abbreviated street types (Rd, Rd., St...)
- Street naming in languages where the street type comes first (Avenue Louis Pasteur)
- Tags that did not represent a street name, representing either something more specific ("ABC Street, 114, 2nd floor") or a region ("Hampshire")
- Inconsistently inserted Zip Codes.

Inconsistently Abbreviated Street Types

The audit.py file from the lectures was adapted for this task. I entered every abbreviation into a mapping dictionary and the unique identifier for each type of street in a "Expected" list. Then I used the following function to update the problematic types:

```
def update_name(name, mapping):

    m = street_type_re.search(name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            name = re.sub(street_type_re, mapping[street_type], name)

    return name
```

Street Naming in Languages Where the Street Type Comes First

This actually only happenned to a couple avenues, and I introduced a mapping changing the tags to the proper translated street type. I initially considered processing the Rio de Janeiro region (where I currently live), and in a context where the name always comes first I would have simply changed the regular expression to match the first word instead of the last.

Tags That Did Not Represent a Street Name

To treat those tags I included a "Ignored" list where I included all such tags and filtered them out in the audit_street_type function

```
def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
```

```
        if (street_type not in expected) and \\
(street_type not in ignored):
            street_types[street_type].add(street_name)
```

Inconsistently Inserted Zip Codes

While the Zip codes in the data set were in better shape than I expected, there were a few that needed correcting. Upon examination there were two problems that occurred: use of the state prefix, breaking the format, and a unknown pattern of numbers which did not respect the format either. To fix this, an auditing function similar to the one made for the street names was created.

Data Overview and Additional Ideas

Here the basic statistics of the dataset are shown as well as the SQL queries used to find them. In the case of the file sizes, the information is obtained in bash.

File Sizes

The bash command "ls -l" returns:

```
fernando 402608128 Jan 19 10:05 boston.db
fernando 743262161 Jan 18 12:17 map
fernando 273287302 Jan 18 17:28 nodes.csv
fernando 29450388 Jan 18 17:28 nodes_tags.csv
fernando 30887311 Jan 18 17:58 ways.csv
fernando 88330571 Jan 18 17:58 ways_nodes.csv
fernando 33908162 Jan 18 17:58 ways_tags.csv
```

Number of Nodes

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

3240071

Number of Ways

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

462112

Number of Unique Users

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

1706

Top 10 Users

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

crschmidt|1348348
jremillard-massgis|1228015
wambag|224440
OceanVortex|132621
morganwahl|126107
Ahlzen|119962
MassGIS Import|108028
ryebread|65834
ingalls_imports|58731
mapper999|15235

Additional Ideas

The contribution seems to have many users who are actually automatic contributors (in the top 10 we see jremillard-massgis and MassGIS Import, I think it's safe to assume that GIS stands for geographical information system, as in ArcGIS). It could be good to sanitize the inputs automatically imported to minimize mistakes. However, breaking the interface that works currently could cause a drop in user contribution.

Along the same line the manual inputs could be sanitized to only accept expected characters. The downside is that in cities where streets are numbered that would be a major inconvenience (such as New York). Also, there are many languages with exotic characters where this input limitation could be a problem. Another possible problem is that the contributors could lose their will to collaborate if their automated tools stop working.

Another possibility would be offering some sort of reward to top non-automated contributors, so that they'd be stimulated. It's also possible to run a cleaner script from time to time to ensure consistency. The downside of running such a script often is that a maintenance team would probably need to be kept to ensure it would perform well. Also, processing all of the information would probably be unfeasible in terms of computational power.

Additional Data

Most Popular Cuisines

Seeing as I'd maybe like to move to Boston, I'm interested in what sort of food I can find more often:

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
...> FROM nodes_tags
...>
JOIN (SELECT DISTINCT(id)
...> FROM nodes_tags WHERE value='restaurant') i
...>
```

```
ON nodes_tags . id=i . id
. . . > WHERE nodes_tags . key=      cuisine
. . . > GROUP BY nodes_tags . value
. . . > ORDER BY num DESC
... > limit 5 ;
```

pizza|51
american|46
chinese|42
mexican|37
italian|36

Most Common Amenities

I figured it's also interesting to know what sort of amenities I might find:

```
 s q l i t e > SELECT value , COUNT(      ) as num
FROM nodes_tags
WHERE key=      amenity
GROUP BY value
ORDER BY num DESC
LIMIT 10 ;
```

bench|1295
restaurant|835
school|597
bicycle_parking|348
place_of_worship|341
library|331
cafe|308
fast_food|265
parking|182
post_box|155

Conclusion

My findings show that while the data has many errors it is reasonably complete and has many contributors. Maybe a more robust import feature and interface with GIS could greatly improve the quality of data. Also it seems that Boston would indeed please me as a place to live with many libraries and italian cuisine.

Sources

I have gathered information to develop this project from the lectures in the Data Wrangling module of the Nanodegree, from the Coursera course named "Introduction to Data Science with Python", from github repositories maintained by users jianru-shi, agapic and alexxucui and also from Stackoverflow, mostly on topic 19472922 regarding the import of the SQL schema into the database via Python.