Report

February 12, 2016

1 Data Analyst Nanodegree OpenStreetMap Project

Author: Brian Novak

1.1 Map Area

Around Baton Rouge, LA, United States.

1.2 Problems Encountered in the Map

1.2.1 Find types of tags, particularly ones with problematic characters.

```
In [1]: run tag_types.py
```

```
{'lower': 70020, 'lower_colon': 121529, 'other': 11680, 'problemchars': 0}

0 tags were found with problematic characters
```

1.2.2 Street Types

The code from lesson 6 of the course was used with some small modifications. This checks for abbreviations in the last word of the street names which is usually the type of street. Some additions were made to the expected street types and the mapping dictionary. When the last word in a street name could be represented as an integer, then that street name was ignored. This ignored 'Highway 42' and 'Highway 44' in this data set.

```
In [2]: run improving_street_names.py
```

```
There are 81 streets in the data set

Street names with abbreviated types:

Essen Ln => Essen Lane
Burbank Dr => Burbank Drive
Juban Rd => Juban Road
Jefferson Hwy => Jefferson Highway
East Parker Blvd => East Parker Boulevard
O'Donovan Blvd => O'Donovan Boulevard
Hazelwood dr => Hazelwood Drive

7 street name abbreviations were found
```

1.2.3 City Names

Since there were only a few cities in the data set, the names were just checked manually and none were found to be spelled incorrectly or abbreviated. Of course if analysis of multiple data sets or larger areas were to be done, it would be worth comparing the city names to a database such as GeoNames (http://download.geonames.org/export/dump/cities1000.zip). The cities in the data set were:

```
In [3]: run find_city_names.py
```

```
Cities included in the data:

Baton Rouge
Livingston
Denham Springs
Central
Walker
Gonzales
Brusly
Port Allen
```

1.2.4 Zip Codes

The zip codes in the addr:postcode fields were checked against the GeoNames zip code data (http://download.geonames.org/export/zip/US.zip), to make sure they were consistent with the city name. No problems were found in the OSM data, but I noticed that the city of Central was missing from the GeoNames data. Central officially became a separate city from Baton Rouge in 2005, but the zip codes in that area remained as 70837. This was not a problem for checking the OSM data since there was only one entry for Central in it and it did not contain the addr:postcode field. Central was also added to the GeoNames data file.

```
In [4]: run check_zip_codes.py
```

```
No problems with zip codes
```

1.2.5 County (parish) id numbers

The GeoNames data also contains the names and id numbers of the counties or parishes in the case of Louisiana and an id number for them. Since the OSM records containing city names sometimes also contain the county id number, the county id numbers in the OSM records were checked to make sure they were consistent with the city names. Records containing both a city name and an id number for the county seem to be rare; there were only two in this data set and both were consistent.

```
In [5]: run check_county_ids.py
```

```
No problems with county id numbers
```

1.2.6 State name abbreviations

Although the state name abbreviations in the addr:state tag are unlikely to be incorrect, it is also easy to do a quick check. Instead of using python, grep can be used to pull out the lines to check and awk can be used to pull out the state abbreviations from the lines. The lack of any output below indicates that all of the state name abbreviations are correct in the addr:state tags.

```
In [6]: %%bash

        cat Baton_Rouge.osm | grep "addr:state" | awk '{print $3}' | awk -F= '{print $2}' \
        | awk -F/ '{print $1}' | awk '$1 !~ "LA"'
```

1.2.7 Consistency of Street Address with City and Latitude and Longitude

Although the zip codes and cities are consistent in the OSM data, there is no guarantee that the street address is not actually in a different city with a different zip code. Although the data set was chosen by location, there is still a possibility that the latitude and longitude are inside the region of interest, but still not consistent with the street address. These things were not checked, but could be checked using reverse geocoding (http://www.geonames.org/export/web-services.html#findNearbyPlaceName, https://developers.google.com/maps/documentation/geocoding/intro?csw=1#ReverseGeocoding).

1.2.8 Apply fixes to street names and convert to JSON to be read into MongoDB

The code from the course was just used on my data set since no additional changes beyond street names were applied.

```
In [7]: run preparing_for_database.py
```

1.2.9 Import data into MongoDB using mongoimport

mongoimport –file Baton_Rouge.osm.json -d OSM -c Baton_Rouge_LA_US_area

1.3 Data Overview

1.3.1 Total number of entries (mongo shell)

db.Baton_Rouge_LA_US_area.count()

249000

1.3.2 Number of entries by city for entries with a city name (mongo shell)

db.Baton_Rouge_LA_US_area.aggregate([{ $match: {"address.city": { $exists: true } } }, { $group: { _id: "$address.city", count: { $sum: 1} } }, { $sort: { "count": -1 } }])

{ "_id" : "Baton Rouge", "count" : 50 }

{ "_id" : "Denham Springs", "count" : 7 }

{ "_id" : "Walker", "count" : 3 }

{ "_id" : "Livingston", "count" : 3 }

{ "_id" : "Port Allen", "count" : 2 }

{ "_id" : "Gonzales", "count" : 2 }

{ "_id" : "Brusly", "count" : 1 }

{ "_id" : "Central", "count" : 1 }

Most of the entries do not contain a city name.

1.3.3 Number of nodes (mongo shell)

db.Baton_Rouge_LA_US_area.find({"type":"node"}).count()

226026

1.3.4 Number of ways (mongo shell)

db.Baton_Rouge_LA_US_area.find({“type”:“node”}).count()

22969

1.3.5 Number of unique users (mongo shell)

db.Baton_Rouge_LA_US_area.distinct(“created.user”).length

322

1.3.6 Counts for top 10 contributing users (mongo shell)

db.Baton_Rouge_LA_US_area.aggregate([{ $group: { _id: “$created.user”, count: { $sum: 1} } }, { $sort: { “count”: -1 } }, { $limit: 10 }])

{ “_id” : “woodpeck_fixbot”, “count” : 62038 }

{ “_id” : “ELadner”, “count” : 28542 }

{ “_id” : “Matt Toups”, “count” : 15694 }

{ “_id” : “Kenneth Pardue”, “count” : 12662 }

{ “_id” : “TIGERcnl”, “count” : 12310 }

{ “_id” : “bot-mode”, “count” : 9613 }

{ “_id” : “25or6to4”, “count” : 8168 }

{ “_id” : “ediyes”, “count” : 7846 }

{ “_id” : “dufekin”, “count” : 7550 }

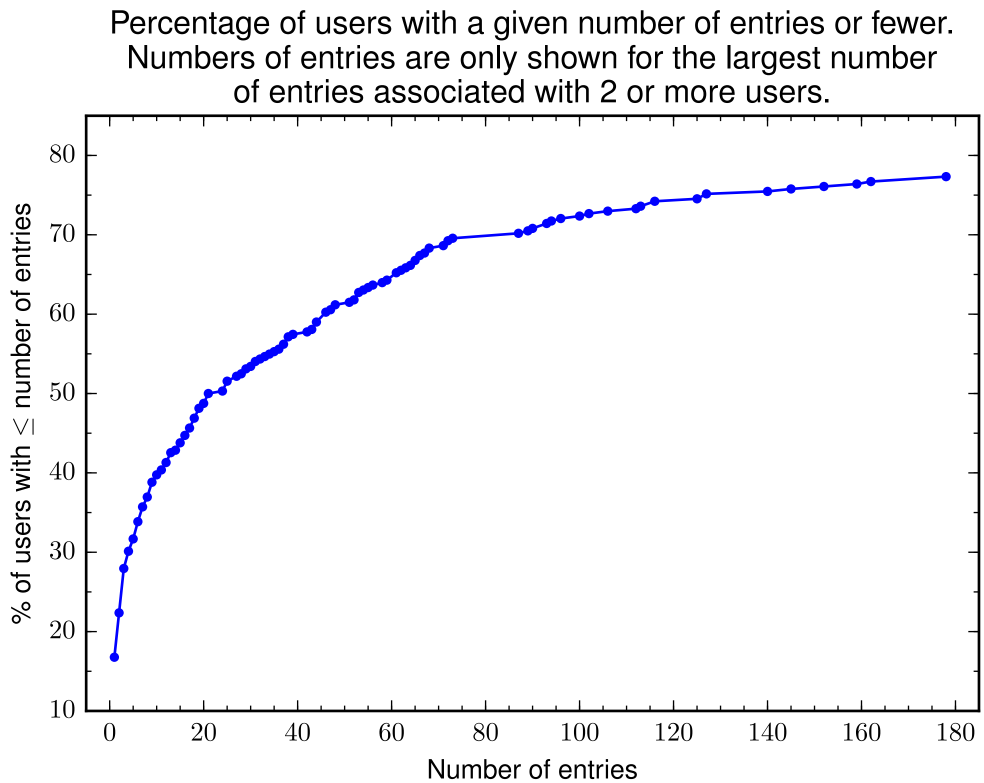
{ “_id” : “rickmastfan67”, “count” : 6771 }

1.3.7 Plot and table of number of users with a given number of entries or fewer as a function of number of entries (pymongo)

```
In [2]: %pylab inline
        %config InlineBackend.figure_format = 'pdf'
        plt.rc('text', usetex=True)
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [3]: run entry_count_hist.py
```



# Entries	# Users	% Users with # Entries or Fewer
1	54	16.7702
2	18	22.3602
3	18	27.9503
4	7	30.1242
5	5	31.677
6	7	33.8509
7	6	35.7143
8	4	36.9565
9	6	38.8199
10	3	39.7516
11	2	40.3727
12	3	41.3043
13	4	42.5466
14	1	42.8571
15	3	43.7888
16	3	44.7205
17	3	45.6522
18	4	46.8944
19	4	48.1366
20	2	48.7578
21	4	50

There are 54 users with only one entry. About 28% of users have 3 or fewer entries, and 50% of users have 21 or fewer entries.

1.4 Additional Ideas or Observations

1.4.1 Users with "bot" or "aut" in their names (mongo shell)

It was noticed that 2 of the 10 users with the largest number of posts had "bot" in their names. These are likely automated entries or fixes. Therefore a search for users with "bot" or "aut" (short for "automatic" or "automated") was performed.

```
db.Baton_Rouge_LA_US_area.aggregate( [ { $match: { "created.user": /.*bot.*/ } }, { $group:
{ _id: "$created.user", count: { $sum: 1} } }, { $sort: { "count": -1 } } ] )
```

```
{ "_id" : "woodpeck_fixbot", "count" : 62038 }

{ "_id" : "bot-mode", "count" : 9613 }

{ "_id" : "xybot", "count" : 19 }
```

```
db.Baton_Rouge_LA_US_area.aggregate( [ { $match: { "created.user": /.*aut.*/ } }, { $group:
{ _id: "$created.user", count: { $sum: 1} } }, { $sort: { "count": -1 } } ] )
```

```
db.Baton_Rouge_LA_US_area.aggregate( [ { $match: { "created.user": /.*bot.*/ } }, { $group:
{ _id: 1, count: { $sum: 1} } } ] )
```

```
{ "_id" : 1, "count" : 71670 }
```

There were 3 users with "bot" in their names, and 0 with "aut" in their names. 71760 of 249000 entries can be contributed to users with "bot" in their names. Therefore it is likely that at least 28.8% of the entries in the data are automated entries or fixes.

1.4.2 Correct street names at the point of entry

It seems that having to make corrections to get consistent street names could be avoided if OSM developed some standards for different countries or regions for how streets should be named. Then names could be checked at the point of entry and users might be asked if the correction is right or the changes could be automatic, although automatic changes would still allow for some errors in cases where the correction is not right.