Project: Wrangle OpenStreetMap Data

Introduction

OpenStreetMap is a community driven project with the goal of providing open-source geographic data for the entire Earth. The purpose of this project will be to take OpenStreetMap data, clean it using data munging techniques, and run SQL queries once the data has been cleaned.

The data I decided to use is from the map of the Island of Hawai'i (also known as the Big Island), in the State of Hawaii, United States of America. I grew up on the Big Island, so it is an area I am familiar with, which should make checking for errors in the data easier.

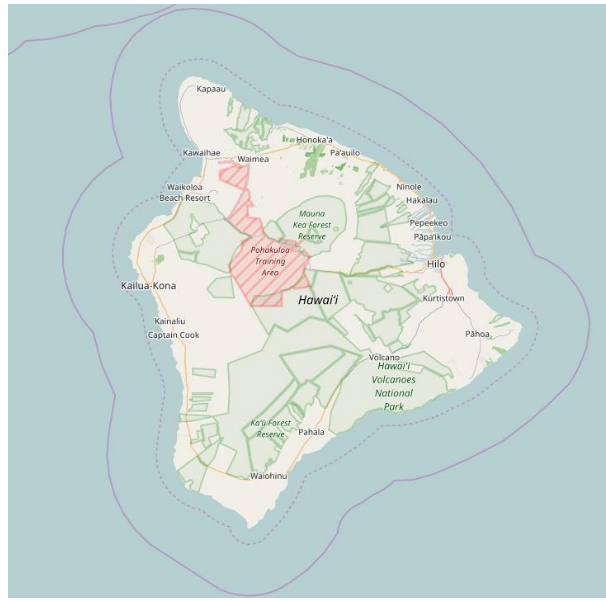


Photo from openstreetmap.org

Problems Encountered in the Map

After auditing a sample of the data using Python, several problems were encountered. Of these, I decided to focus on cleaning the street name, zip code, and telephone number data:

Street Names

The first issue involving the street names was that they did not adhere to a standard format:

- Some street names used abbreviations in their names while others do not (e.g. Milo St. vs. Lako Street or Puni Lapa Loop N. vs. Puni Mauka Loop North)
- Some names contain words in all lowercase (e.g. Kapoi street)

Secondly, way elements for street data have tags containing the street name where the key is "*name*" instead of "*addr:street*."

To clean the street names data, I modified the "update_name" function from the Udacity Case Study Lesson. After making changes, the function will: Unabbreviate all abbreviated words, change words written in all lowercase to have the first letter capitalized, and update the tag key to be "street."

Zip Codes

I found that there were no problems with how the zip codes were formatted, however, three zip codes were erroneous. One of the zip codes was is a zip code used on the island of Moloka'i, the other is a zip code that is an Alaskan zip code, and the last was given as "HI."

The after looking up the true zip codes for each of the three locations with inconsistent codes, I created the "update_zip" function to correct them.

Phone Numbers

There were 15 different formats for how phone numbers were formatted. Here are some examples:

- (808) XXX – XXXX
- (808) – XXX – XXXX
- +1 808 XXX XXXX
- +1 808 XXX – XXXX
- +1 (808) XXXXXXX
- +1 – 808 – XXX – XXXX
- XXXXXXX

The "update_phones" function converts all phone numbers to the format

(808) – XXX – XXXX.

Overview of the Data

After writing Python code to clean the data, the updated data was written to csv files and then loaded into a database. Here are the file sizes for the original OpenStreetMap XML file as well as the csv files, found using Python:

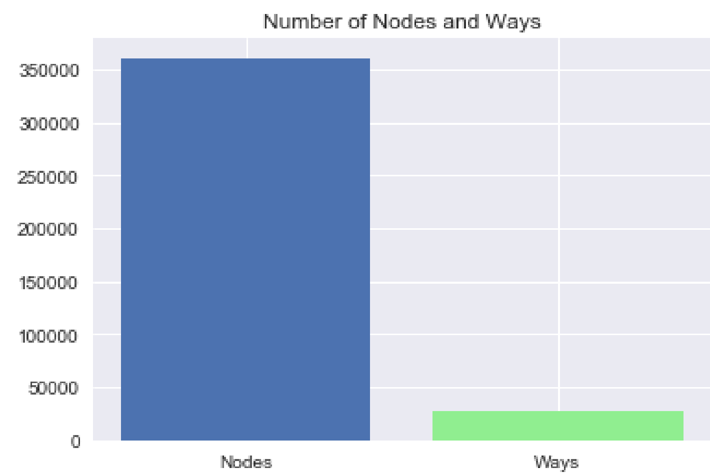
```
County_of_Hawaii.osm...................: 69M
hawaii_county.db.......................: 41M
nodes.csv..............................: 29M
nodes_tags.csv.........................: 396K
sample.osm.............................: 7M
ways.csv...............................: 1M
ways_nodes.csv.........................: 9M
ways_tags.csv..........................: 2M
```

At this point, I had come up with a few questions that I wanted to answer by querying the database using SQlite. Below are these questions, the queries I performed using Python with the SQLite3 module, along with the results:

How Do the Number of Nodes and Ways Compare?

```
QUERY = 'SELECT COUNT(*) \
         FROM node;'

number_of_nodes = c.execute(QUERY).fetchall()

print 'Number of Nodes:  {}'.format(number_of_nodes[0][0])

QUERY = 'SELECT COUNT(*) \
         FROM way;'

number_of_ways = c.execute(QUERY).fetchall()

print 'Number of Ways:  {}'.format(number_of_ways[0][0])
```

```
Number of Nodes:  360720
Number of Ways:  26775
```



How Many Tourist Attractions are on the Big Island?

```
QUERY = 'SELECT COUNT(*) \
         FROM (SELECT * FROM node_tags UNION ALL SELECT * FROM way_tags)
all_tags \
         WHERE all_tags.key == "tourism" and all_tags.value == "attraction";'

tourism = c.execute(QUERY).fetchall()

print ''
print 'Number of Tourism Attractions:  {}'.format(tourism[0][0])
```

```
Number of Tourism Attractions:  40
```

How Many Historic Sites are on the Big Island?

```
QUERY = 'SELECT COUNT(*) \
         FROM (SELECT * FROM node_tags UNION ALL SELECT * FROM way_tags)
all_tags \
         WHERE all_tags.key == "historic";'

historic = c.execute(QUERY).fetchall()

print ''
print 'Number of Historic Sites:  {}'.format(historic[0][0])
```

```
Number of Historic Sites:  25
```

How Many Beaches are on the Big Island?

```
QUERY = 'SELECT COUNT(*) \
         FROM (SELECT * FROM node_tags UNION ALL SELECT * FROM way_tags)
all_tags \
         WHERE all_tags.key == "natural" and all_tags.value == "beach";'

beaches = c.execute(QUERY).fetchall()

print ''
print 'Number of Beaches:  {}'.format(beaches[0][0])
```

```
Number of Beaches:  58
```

What is the Frequency for Each Zip Code on the Big Island?

```
QUERY = 'SELECT all_tags.value, COUNT(*) as num \
         FROM (SELECT * FROM node_tags UNION ALL SELECT * FROM way_tags)
all_tags \
         WHERE all_tags.key == "postcode" \
         GROUP BY all_tags.value \
         ORDER BY num DESC'

zip_codes = pd.read_sql_query(QUERY, conn)
zip_codes.columns = ['Zip Code', 'Frequency']
zip_codes.index = np.arange(1, len(zip_codes) + 1)

print ''
print 'Frequency of Zip Codes'
print zip_codes
```

```
Frequency of Zip Codes
   Zip Code  Frequency
1     96778         58
2     96720         37
3     96740         32
4     96727          6
```

```
5     96749          5
6     96704          4
7     96710          4
8     96738          4
9     96743          4
10    96737          2
11    96755          2
12    96771          2
13    96772          2
14    96777          2
15    96785          2
16    96725          1
17    96750          1
18    96760          1
19    96783          1
```

The results from this query surprised me since the zip codes for the Big Island's two largest towns Hilo, and Kona are second and third on this list respectively. Also, there appear to be a relatively high amount of occurrences for the zip code "96778." However, it appears that overall there is not a lot of zip code data, so perhaps 96778 is simply overrepresented in the dataset while 96720 and 96740 are underrepresented.

What are the Number of Unique Users?

```
QUERY = 'SELECT COUNT(DISTINCT(nodes_and_ways.user)) \
         FROM (SELECT user FROM node UNION ALL SELECT user FROM way)
nodes_and_ways;'

users = c.execute(QUERY).fetchall()

print ''
print 'Number of unique users:  {}'.format(users[0][0])

Number of unique users:  314
```

Who are the Top 10 Most Prolific Users?

```
QUERY = 'SELECT nodes_and_ways.user as unq_user, COUNT(*) as num \
         FROM (SELECT user FROM node UNION ALL SELECT user FROM way)
nodes_and_ways \
         GROUP BY unq_user \
         ORDER BY num DESC \
         LIMIT 10;'

top_users = pd.read_sql_query(QUERY, conn)
top_users.columns = ['User Name', 'Frequency']
top_users.index = np.arange(1, len(top_users) + 1)

print ''
print 'Top 10 Users'
print top_users
```

```
Top 10 Users
                         User Name  Frequency
1                      Tom_Holland     144965
2                         bdiscoe     104865
3                        ksamples      66860
4                  Chris Lawrence       6420
5                        monaliki       6240
6                            Vlad       5184
7    Mission Aware Technologies        4960
8                       InfiNorth       4553
9                            dima       3534
10                        OklaNHD       3454
```

What are the Top 10 Most Frequent Amenities?

```
QUERY = 'SELECT value, COUNT(*) as num \
         FROM (SELECT * FROM node_tags UNION ALL SELECT * FROM way_tags)
all_tags \
         WHERE all_tags.key == "amenity" \
         GROUP BY value \
         ORDER BY num DESC \
         LIMIT 10'

top_amenities = pd.read_sql_query(QUERY, conn)
top_amenities.columns = ['Amenity', 'Frequency']
top_amenities.index = np.arange(1, len(top_amenities) + 1)

print ''
print 'Top 10 Amenities'
print top_amenities
```

```
Top 10 Amenities
             Amenity  Frequency
1            parking        427
2         restaurant        107
3            toilets         69
4               fuel         45
5          fast_food         35
6               cafe         32
7             school         32
8    place_of_worship        31
9          recycling         18
10       post_office         17
```

What are the Top 10 Most Frequent Restaurant Cuisines?

```
QUERY = 'SELECT value, COUNT(*) as num \
         FROM node_tags \
         WHERE node_tags.key == "cuisine" \
         GROUP BY value \
         ORDER BY num DESC \
```

```
         LIMIT 10'

top_cuisines = pd.read_sql_query(QUERY, conn)
top_cuisines.columns = ['Cuisine', 'Frequency']
top_cuisines.index = np.arange(1, len(top_cuisines) + 1)

print ''
print 'Top 10 Cuisines'
print top_cuisines

Top 10 Cuisines
      Cuisine  Frequency
1        thai          8
2      burger          6
3       pizza          6
4    american          4
5     chinese          4
6   ice_cream          4
7     italian          4
8    japanese          4
9     mexican          4
10   regional          4
```

There are many Thai restaurants in my hometown of Kona, so Thai being the most frequent cuisine is not surprising here.

Other Ideas about the Dataset

From the database query for zip codes, it is apparent there is not much zip code data and that many zip codes are missing from the dataset. The Big Island has 32 total zip codes, yet only 19 are listed in the database. Ideally, data currently missing zip codes could have zip codes added manually by users or programmatically by using latitude and longitude or address data to find the correct zip code.

Another possible shortcoming with the data is that for tags with the amenity key, there are separate values that are very similar such as “restaurant”, “fast_food”, and “cafe.” If these values were grouped together under a new category such as “food”, it would reduce the amount of values one would have to keep track of in the data. However, if we did group these together under a general “food” category, this wouldn’t be helpful if one wanted to query for detailed information, such as the number of fast food restaurants, therefore this is a bit of a gray area. A possible solution would be to create a Python dictionary for each value in tags for amenities, saving the general category as the dictionary key and the detailed value as the dictionary value.

Conclusion

After the auditing and cleaning the OpenStreetMap data for the Big Island is certainly in a better shape than when it was downloaded. The street names and telephones now have a standard format, street names have a tag with key: “street”, and zip codes that were erroneous were corrected. That being said, it is still far from perfect as there are missing zip code data as well as tag values that could be more general.

References

https://mapzen.com/data/metro-extracts/your-extracts/7fd309dd6da8

https://stackoverflow.com/questions/740287/how-to-check-if-one-of-the-following-items-is-in-a-list

https://www.tutorialspoint.com/python/string_replace.htm

https://stackoverflow.com/questions/3728655/titlecasing-a-string-with-exceptions

https://stackoverflow.com/questions/9222106/how-to-extract-information-between-two-unique-words-in-a-large-text-file

https://stackoverflow.com/questions/10365225/extract-digits-in-a-simple-way-from-a-python-string

http://pandas.pydata.org/pandas-docs/version/0.20/generated/pandas.read_sql_query.html#pandas.read_sql_query

http://sebastianraschka.com/Articles/2014_sqlite_in_python_tutorial.html

https://www.openstreetmap.org/about

https://www.openstreetmap.org/#map=8/19.950/-156.962

https://discussions.udacity.com/t/display-files-and-their-sizes-in-directory/186741/2