SF_OpenStreetMap_project

November 18, 2017

This project is part of Udacity's Data Analysis Nano Degree Project. Below are the steps I followed for this project:

1. Choosing a map area from OpenStreetMap and downloading the xml file of the chosen area.
2. Auditing the accuracy, uniformity, consistency and completeness of the xml data.
3. Cleaning the dataset based on the result of the audit.
4. Storing the cleaned dataset into SQLite database.
5. Querying interesting features for the chosen map.

0.1 Map Area

San Francisco/Bay Area, California, USA

I chose the San Francisco/Bay Area map because I recently moved to San Francisco and I am eager to learn more about my city.

1 Problems Encountered

1. Abbreviated street names

- For abbreviated street names such as Ave, Blvd, and Cr, I corrected those street names into non-abbreviated forms.

2. Inconsistency of zipcode format

- For zipcodes with two strings of numbers with a hyphen in between (95004-9506), I removed the numbers after the hyphen as this refers to a more specific location within a given zipcode.
- For zipcodes with state abbreviations i.e. 'CA' in the beginning (CA 95110), I removed the abbreviation and kept only the number string.

3. Inconsistency of city name format

- For city names in lower case ('sacramento'), I capitalized the first letter.
- For city names with all caps ('MORGAN HILL'), I capitalized only the first letter.
- For city names with the abbreviation 'CA' ('Sacramento, CA'), I removed the "CA" string

After performing the data cleaning and extracting the query for the city names, I found that a number of city names were still accompanied with the state abbreviation 'CA' such as 'Alameda, CA.' It turned out that the city names stored in ways_tags were collected from the TIGER dataset, a third party dataset. Setting ‘tiger:county’ as a key, I performed the following function below to determine whether the state abbreviation appeared uniformly across the dataset.

```
In [ ]: import xml.etree.cElementTree as ET
        import pprint
        file = 'small_sample.osm'

        def get_zipcode(filename):
                tag_type = set()
                for event, elem in ET.iterparse(filename):
                        if elem.tag == 'node' or elem.tag == 'way':
                                for tag in elem.iter('tag'):
                                        if tag.attrib['k'] == 'tiger:county':
                                                tag_type.add(tag.attrib['v'])
                return tag_type
```

The above function produced the following result below, proving that the string ‘CA’ was featured uniformly across the TIGER dataset.

```
In [ ]: set(['Alameda, CA',
             'Contra Costa, CA',
             'El Dorado, CA',
             'Lake, CA',
             'Marin, CA',
             'Merced, CA',
             'Monterey, CA',
             'Napa, CA',
             'Placer, CA',
             'Sacramento, CA',
             'San Benito, CA',
             'San Francisco, CA',
             'San Joaquin, CA',
             'San Mateo, CA',
             'San Mateo, CA;Santa Clara, CA',
             'Santa Clara, CA',
             'Santa Cruz, CA',
             'Solano, CA',
             'Solano, CA; Napa, CA',
             'Sonoma, CA',
             'Stanislaus, CA',
             'Yolo, CA'])
```

I then added the function below to the main data script, thus cleaning the data to extract the string ‘CA’ from ‘tiger:county’ keys.

```
In [ ]: def tiger_county(data):
                if ';' in data:
                        f = data.split(';')[0]
                        data = f.split(', ')[0]
                else:
                        data = data.split(', ')[0]
                return data
```

2 Data Overview

2.0.1 File Size

```
In [ ]: nodes.csv...............................: 24M
        nodes_tags.csv..........................: 497K
        sf.db...................................: 34M
        sf.osm..................................: 63M
        ways.csv................................: 2M
        ways_nodes.csv..........................: 7M
        ways_tags.csv...........................: 3M
```

2.0.2 Number of Nodes

```
In [ ]: SELECT COUNT(*) FROM nodes;
```

318731

2.0.3 Number of Ways

```
In [ ]: SELECT COUNT(*) FROM ways;
```

35617

2.0.4 Number of Unique Users

```
In [ ]: SELECT COUNT(DISTINCT(subq.uid)) FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM
```

2314

2.0.5 Top 10 Contributors

```
In [ ]: SELECT subq.user, COUNT(*) AS num
        FROM (select user, uid from nodes union all select user, uid from ways) subq
        GROUP BY subq.uid
        ORDER BY num DESC
        LIMIT 10;
```

```
In [ ]: andygol|37126
        nmixter|33785
        ediyes|21395
        woodpeck_fixbot|17774
        Luis36995|15851
        Eureka gold|15619
        dannykath|14252
        RichRico|12917
        Rub21|7997
        mk408|7330
```

2.0.6 Top 10 Appearing Cities

```
In [ ]: SELECT e.value, COUNT(*) AS num
        FROM (select key, value from nodes_tags union all select key, value from ways_tags) e
        WHERE e.key = 'city'
        GROUP BY e.value
        ORDER BY num DESC
        LIMIT 10;
```

```
In [ ]: Stockton|1362
        Redwood City|475
        West Sacramento|413
        Palo Alto|388
        San Francisco|381
        Berkeley|124
        Hollister|115
        Sunnyvale|87
        Piedmont|82
        Mountain View|56
```

2.0.7 Top 10 Appearing Fast Food Chains Restaurants

```
In [ ]: SELECT value, COUNT(*) AS num
        FROM nodes_tags
        JOIN (select distinct(id) from nodes_tags where key = 'cuisine') subq
        ON nodes_tags.id = subq.id
        WHERE nodes_tags.key = 'name'
        GROUP BY nodes_tags.value
        ORDER BY num DESC
        LIMIT 10;
```

```
In [ ]: Panda Express|3
        Starbucks|3
        Cactus Taqueria|2
        Chipotle|2
        Eureka!|2
        Five Guys|2
        Il Postale|2
        KFC|2
        Nation's Giant Hamburgers|2
        Nico's 1508|2
```

2.0.8 Top 10 Appearing Coffee Stores

```
In [ ]: SELECT nodes_tags.value, COUNT(*) AS num
        FROM nodes_tags
        JOIN (select distinct(id) from nodes_tags where value = 'cafe') subq
        ON nodes_tags.id = subq.id
        WHERE nodes_tags.key = 'name'
        GROUP BY nodes_tags.value
        ORDER BY num DESC
        LIMIT 10;
```

```
In [ ]: Starbucks|7
        Peet's Coffee & Tea|2
        Bridgeway Cafe|1
        BroomBush Cafe|1
        Chromatic Cafe|1
        College Point Cafe|1
        Comforts|1
        Contraband Coffee Bar|1
        East Bay Coffee Co.|1
        Eva's coffee|1
```

2.0.9 Top 10 Appearing Cuisines

```
In [ ]: SELECT value, COUNT(*) AS num
        FROM (select key, value from nodes_tags union all select key, value from ways_tags) e
        WHERE e.key = 'cuisine'
        GROUP BY e.value
        ORDER BY num DESC;
```

```
In [ ]: burger|22
        mexican|22
        chinese|13
        pizza|13
        coffee_shop|11
        thai|10
        american|8
        italian|6
        sandwich|6
        vietnamese|6
```

3 Other Ideas about the Dataset

3.0.1 Social Media Integration in the OpenStreetMap Dataset

- As I was exploring the number of nodes for ATM in the San Francisco/Bay Area, I noticed that there was only one Bank of America ATM in the dataset. Although the dataset is only 63 megabytes out of the 3.28 gigabyte dataset, this seems disproportionate for the large number of ATMs in any major city. In order to capture a more accurate picture of the number of

ATMs in the city, perhaps social media e.g. Facebook and Twitter could be the solution. A Facebook page or Twitter handle could be created for the ATM by banks, enabling ATMs to be incorporated into OpenStreetMap location data. By creating a system that connects Facebook and Twitter's API data with the OpenStreetMap dataset, the OpenStreetMap database could collect more location data for service locations such as ATMs.

- However, integrating the OpenStreetMap Dataset with social media may result in varying data formats inputted by social media users making it harder to maintain the quality of data in terms of consistency and accuracy. This might result in the need for extensive data cleaning which could be costly.

4 Conclusion

I anticipated that the San Francisco/Bay Area OpenStreetMap dataset would be clean, however I was able to find a number of messy records which required cleaning. Some business locations, such as ATMs and coffee shops were lacking from the geographical data. It would be interesting to see how the OpenStreetMap Dataset could evolve to capture these additional business locations. ATMs and coffee shops are indeed an essential part of San Francisco life!

5 References

- https://mapzen.com/data/metro-extracts/metro/san-francisco-bay_california/
- https://www.census.gov/geo/maps-data/data/tiger.html
- http://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Custom%20Keyboard%20Shortc