OpenStreetMap Data Case Study

Roger Duong October 2017

Map Area

Singapore

- https://www.openstreetmap.org/export#map=12/1.3450/103.8500

The map used is the city I currently live in.

Sources

Singapore Postal Code system

- http://eresources.nlb.gov.sg/infopedia/articles/SIP_1006_2010-05-27.html

Workflow

This project is organized with the following workflow:

1. Launch the audit scripts to explore the dataset (`audit.py`)
2. Identify problems encountered in the map, develop programmatic ways to solve those issues and include them into `transform.py`
3. Extract the data from the osm dataset (`load.py`)
4. Transform the extracted dataset by calling (`transform.py`)
5. Load the transformed dataset into csv files (`load.py`)
6. Load the csv files into a sqlite3 database (`schema.sql`)
7. Run SQL queries to explore the map dataset

Problems Encountered in the Map

This section we discuss the findings of the step 2 of the workflow. After extracting a sample of the entire dataset, and running it through an audit.py file, we can notice the following issues:

1. Incorrect street types.
2. Incorrect postal codes.

The following elaborates on the issues encountered.

Street Types

The audit script revealed several types of incorrect street types:

1. Overabbreviated street names: for example 'St' instead of 'Street'.
2. Incorrectly capitalized street names: for example 'road' instead of 'Road'.
3. Typo errors: for example 'aenue' instead of 'Avenue'.

It should be noted that in Singapore street types may appear:

- At the end of the street name, like in the U.S. For example 'Orchard *Boulevard*'.
- At the beginning of the street name. For example '*Jalan* Besar'.
- At the second to last position in the street name. For example 'Ang Mo Kio *Avenue* 1' (note: the '1' is not the house number, it is the street number, as there are 10 street named 'Ang Mo Kio Avenue 1' to 'Ang Mo Kio Avenue 10').

The helper function `audit_street_type` in audit.py takes into account those locale specifics.

```
mapping = {
    'Lor' : 'Lorong',
    'Terrance' : 'Terrace',
    'terrace' : 'Terrace',
    'St' : 'Street',
    'Upp' : 'Upper',
    'road' : 'Road',
    'Rd' : 'Road',
    ' rd' : ' Road',
    'Roadc' : 'Road',
    'Roads' : 'Road',
    'Pl' : 'Place',
    'Jln' : 'Jalan',
    'aenue' : 'Avenue',
    'Ave' : 'Avenue',
    'Avebue' : 'Avenue',
    'AveNue' : 'Avenue',
    'avenue' : 'Avenue',
    'Blvd' : 'Boulevard',
    'Dr' : 'Drive',
    'drive' : 'Drive',
    'garden' : 'Garden',
    'geylang' : 'Geylang',
    'park' : 'Park',
}

def clean_street_name(street_name, mapping):
    """Clean street name, by replacing dictionary strings found
```

```
    in street_name in mapping"""

    m_end = audit.street_type_end.search(street_name)
    m_num = audit.street_type_num.search(street_name)
    street_type = ''
    if m_end:
        street_type = m_end.group()
    elif m_num:
        if len(re.split(' ', street_name)) >=2:
            street_type = re.split(' ', street_name)[-2]
        else:
            street_type = re.split(' ', street_name)[-1]

    if street_type in mapping.keys():
        new_street_name = street_name.replace(street_type, mapping[street_type])
        print(street_name+ ' --> ' + new_street_name)
        return(new_street_name)
```

Postal Codes

Singapore uses a postal code system where each postal delivery point –usually each building– is identified with a unique 6-digit postal code. Therefore entries with anything other than 6 digits are incorrect.

Initial audit of the data revealed the following types of incorrect entries:

1. Entries where the last characters are the postal code, and where the leading character strings are any kind of text, usually 'Singapore 123456'. Solving these incorrect entries simply consist in removing all space characters and selecting the sub-string of the last 6 characters.
2. Entries of 5 characters, where the leading zero of the correct entry has been stripped down, probably by casting the postal code to an integer type. Solving these incorrect entries require only to add the leading zero.
3. Other incorrect entries not falling into the categories above. Solving the incorrect entries can be done by querying the Singapore Post website, which has a page to search for postal codes for a given house number and street. This required a helper function to scrape the webpage and to retrieve the data. The helper function to get the postal code incorporate some time delay and user-agent parameters to avoid being blocked by the Singapore Post website after repeated queries. Because of the time delay, it was important to treat the highest number of incorrect entries by the methods 1 and 2 above, to avoid slow script execution.

```
def get_postal_code(html_page, house_number, street_name):
    """Get Postal Code from web page of Singapore Post"""
```

```
        time.sleep(5) #Delay start of scraping for 5 seconds

        headers = {'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWebKit/5
        data = {'building': house_number,
                'street_name': street_name}

        r = requests.post(html_page, data=data, headers=headers, timeout=2)

        c = r.content
        soup = BeautifulSoup(c, "lxml")

        result = soup.find(id = 'datatable-1')

        if result is not None: #scrape through the result page to datatable-1
            for val in result.findAll('p'):
                if len(val.text) == 6:
                    postal_code = val.text
            return postal_code
        else:
            return('Error')

    def clean_postal_code(old_postal_code, house_number, street_name):
        """Clean postal code, by replacing old_postal_code with new name retrieved
        from Singapore Post"""

        if len(old_postal_code) != 6:
            if ' ' in old_postal_code: #to account for postal codes like 'Singapore 123456'
                new_postal_code = old_postal_code.replace(' ','')[-6:]
            elif len(old_postal_code) == 5: #account for leading zeros being removed
                new_postal_code = '0' + old_postal_code
            else: #get the postal code by querying the Singapore Post website
                get_result = get_postal_code(postal_code_html_page, house_number, street_name)
                if get_result != 'Error':
                    new_postal_code = get_result
                    print('Successfully retrieved postal code from Singpost for')
                else:
                    new_postal_code = old_postal_code
                    print('Error retrieving postal code from Singpost for')
            print(house_number + ' ' + street_name)
            print(old_postal_code + ' --> ' + str(new_postal_code))
            return(new_postal_code)
        else:
            return(old_postal_code)
```

Data Overview

This section discusses the findings of the setp 7 of the project workflow. It describes basic statistics about the dataset.

File sizes

The summary of file size is returned by a helper function in the load.py file.

```
def summarize_dataset(dataset):
    """Return summary of dataset filesize"""
    for file in dataset:
        filename = re.split('/', file)[-1]
        print(filename + '\t{0:.2f} MB'.format(os.path.getsize(file)/(10**6)))
```

```
singapore.osm   211.41 MB
nodes.csv       73.31 MB
nodes_tags.csv  3.54 MB
ways.csv        8.66 MB
ways_nodes.csv  26.80 MB
ways_tags.csv   13.55 MB
```

Number of nodes

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

```
887137
```

Number of ways

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

```
144159
```

Number of unique users

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

```
1520
```

Top 10 contributing users

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

```
JaLooNz,275589
cboothroyd,50194
Luis36995,38471
ridixcr,38004
calfarome,32845
rene78,29926
nikhilprabhakar,22755
yurasi,20454
jaredc,19039
dmastin82,16963
```

Data Exploration

This section further drills down on the dataset. It discusses some findings of step 7 of the project workflow.

Top 10 appearing amenities

Let's start the exploration by looking at the most frequent amenities listed in this map.

```
SELECT tags.key, tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = "amenity"
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

```
amenity,restaurant,1755
amenity,parking,1689
amenity,atm,702
amenity,cafe,459
amenity,school,458
amenity,place_of_worship,381
amenity,fast_food,315
amenity,taxi,310
amenity,bank,230
amenity,swimming_pool,230
```

Total number of restaurants

From the previous query, we can see that there are lots of restaurants. This is not surprising as in Singapore, we love to eat out. So let's show some statistics about our restaurants.

```
SELECT COUNT(*) as count
FROM nodes_tags
    JOIN (SELECT DISTINCT(id)
    FROM nodes_tags
    WHERE nodes_tags.value = "restaurant") as nt
    ON nt.id = nodes_tags.id
WHERE nodes_tags.key = "cuisine";
```

```
636
```

List top 20 cuisines of restaurants

Let's further drill down by returning the top most frequent cuisines of restaurants.

```
SELECT nodes_tags.value, COUNT(*) as count
FROM nodes_tags
    JOIN (SELECT DISTINCT(id)
    FROM nodes_tags
    WHERE nodes_tags.value = "restaurant") as nt
    ON nt.id = nodes_tags.id
WHERE nodes_tags.key = "cuisine"
GROUP BY nodes_tags.value
ORDER BY count DESC
LIMIT 20;
```

```
chinese,135
japanese,72
korean,44
pizza,43
italian,37
indian,35
asian,31
thai,29
french,15
seafood,13
burger,12
international,9
regional,9
vegetarian,6
vietnamese,6
western,6
american,5
chicken,5
indonesian,5
steak_house,5
```

As we can see, although Chinese cuisine –unsurprisingly– dominates the types of cuisines, there is a very wide variety of cuisines. This truly makes Singapore a cosmopolitain city!

Total number of cafes

Here we take a look at the cafes in Singapore.

```
SELECT COUNT(*) as count
FROM nodes_tags
    JOIN (SELECT DISTINCT(id)
    FROM nodes_tags
    WHERE nodes_tags.value = "cafe") as nt
    ON nt.id = nodes_tags.id
WHERE nodes_tags.key = "cuisine";
```

```
91
```

List top 20 styles of cafe

We further drill down with the style of cafes.

```
SELECT nodes_tags.value, COUNT(*) as count
```

```
FROM nodes_tags
    JOIN (SELECT DISTINCT(id)
    FROM nodes_tags
    WHERE nodes_tags.value = "cafe") as nt
    ON nt.id = nodes_tags.id
WHERE nodes_tags.key = "cuisine"
GROUP BY nodes_tags.value
ORDER BY count DESC
LIMIT 20;
```

```
coffee_shop,43
international,6
regional,4
sandwich,4
italian,3
Western,2
asian,2
coffee_shop;regional,2
french,2
"Hawker or Foodcourt, Chinese",1
Nanyang_Coffee,1
Western/Italian,1
acai,1
american;italian_pizza,1
breakfast;coffee_shop,1
cafe,1
cafe/diner,1
cake,1
chicken,1
coffee_shop;coffee,1
```

Looking at the list of styles for cafes, it appears that the entries should be further cleaned:

- some entries contain multiple categories separated by a semi-colon or a slash like: `coffee_shop;regional`
- some entries have inconsistent capitalization like: `Western` and `asian`
- some entries are registered as strings within quotation marks like `"Hawker or Foodcourt, Chinese"`

List top 20 styles of fast-foods

We run the same type of analysis for the fast-foods.

```
SELECT nodes_tags.value, COUNT(*) as count
```

```
FROM nodes_tags
JOIN (SELECT DISTINCT(id)
FROM nodes_tags
WHERE nodes_tags.value = "fast_food") as nt
ON nt.id = nodes_tags.id
WHERE nodes_tags.key = "cuisine"
GROUP BY nodes_tags.value
ORDER BY count DESC
LIMIT 20;
```

```
burger,59
chicken,27
sandwich,13
pizza,12
chinese,7
fast_food,5
ice_cream,5
asian,4
american,2
japanese,2
kebab,2
regional,2
"Curry Puffs",1
Fried_Chicken,1
Hawker_Centre,1
Sandwich,1
american;burger,1
burger;japanese,1
coffee_shop,1
fish;burger;breakfast;ice_cream;tea;cake;coffee_shop;american;chicken,1
```

This list of nodes tags shows the same data quality issues as the nodes tags on cafes.

List of top 10 sports

Singapore has the reputation of being a great city for active people. So let's take a look at the sport amenities and other outdoor facilities for leisure.

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
  SELECT * FROM ways_tags) tags
WHERE tags.key = "sport"
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

```
tennis,358
swimming,302
basketball,114
soccer,81
golf,32
multi,16
badminton,12
running,12
equestrian,7
yoga,7
```

List of top 10 leisure

```sql
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = "leisure"
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

```
swimming_pool,944
pitch,804
park,475
playground,211
park_connector,80
sports_centre,72
fitness_centre,58
garden,41
fitness_station,33
recreation_ground,23
```

The list of leisure facilities appears cleaner than the list of food & beverage locations (restaurants, cafes and fast-foods).

One explanation can be that entries for restaurants business are changing more frequently that entries for sport facilities. It is common to see restaurants open and close within 2 years. Because of this higher frequency, user-defined information about restaurant businesses have a higher propensity to have inconsistent data.

Another explanation is that information on food & beverage amenities is by nature harder to classify than sports and leisure amenities. A basketball court is easily identified, while a cafe serving pizzas and chinese noodles is more challenging to categorize (and surprisingly, there

are such places...).

Conclusion

This data wrangling exercice on Singapore Open Street Map data highlights that there is a great wealth of information available for further analysis. However there is room for further cleaning, especially in the fields of food and beverage location information.

With some further scripting, it will be feasible to clean up some of this information.