OpenStreetMap Data Project

Map Area

Hong Kong, China

- https://www.openstreetmap.org/node/2833125787
- https://mapzen.com/data/metro-extracts/

This is an amazing city where I am living with my family. I have been staying here for more than ten years, so I'm interested to see what database querying reveals, and I would like to take this opportunity to contribute to its improvement on OpenStreetMap.org.

Problems Encountered in the Map

After initially downloading a small sample size file and running it against provisional audit.py and data.py files, I noticed three main problems with the data, which I will discuss in the following order:

- Overabbreviated street names *("d'Aguilar St")*
- "Incorrect" postal codes *(No zip code for Hong Kong, but a small count of numbers begin with "51" indicate areas outside this region.)*
- "Incorrect" phone numbers (*Country code for Hong Kong should be "+852", but a small count of other numbers besides "+852" or just absent.*)

Overabbreviated Street Names

Once the data was imported to SQL, some basic querying revealed street name abbreviations and postal code incorrect. To deal with the over abbreviations, I tried to iterate over each word in an address, correcting them to their respective mappings in audit.py using the following function:

```
def update_name(name, mapping):
    words = name.split()
    for w in range(len(words)):
        if words[w] in mapping:
           #print words[w]
           words[w] = mapping[words[w]]
           name = " ".join(words)
    return name
```

This updated all substrings in problematic address strings, such that: *“1 Stewart Rd”* becomes: *“1 Stewart Road”; “d'Aguilar St”* becomes: *“d’Aguilar Street”.*

Postal Codes

Because Hong Kong is a Special Administrative Region, no postal code is available for this place. However, SQL query still returned some inputs as “incorrect postal code”:

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

Here are the top ten results, beginning with the highest count:

```
value,count
518000,11
510623,7
510180,6
519000,6
510290,5
518048,5
519087,5
518038,4
518067,4
"QBML 2”,4
```

From the postal codes beginning with the number of 510, I suspected that the mistakes might come from data extracted from GPS, which miscounted neighboring cities in mainland China as part of Hong Kong, such as Shenzhen and Zhuhai. To explore this, I performed another aggregation to verify a certain suspicion...

Sort cities by count, descending

```
sqlite> SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key LIKE '%city'
```

```
GROUP BY tags.value
ORDER BY count DESC
LIMIT 10;
```

And, the results, edited for readability:

```
"香港 Hong Kong",363
"屯門 Tuen Mun",128
"荃灣 Tsuen Wan",75
"Sai Kung",53
"广东省深圳市",50
"Hong Kong",49
"Ta Kwu Ling",41
"深圳",40
Zhuhai,37
Shenzhen,31
```

Besides Hong Kong, in these top 10 results, only Tuen Mun, Tsuen Wan, Ta Kwu Ling can be considered as a part of Hong Kong. These ‘cities’ might be mis-labeled and assigned a postal code. However, the rest cities, including Shenzhen and Zhuhai, are all neighboring areas. These results confirmed my suspicion that this metro extract would perhaps also include surrounding cities near Hong Kong. This is hard to be cleaned programmatically, since that the coordinates of ‘cities’ with a postal code assigned should be checked manually, to distinguish whether it is a part of Hong Kong or just an neighboring city.

Phone Numbers

The intentional country code for Hong Kong is “+852”. However, some basic SQL querying revealed only a small count of phone numbers starting with “+852”. The international country code for some numbers are absent, while some are coded as “+853” or others:

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

And the results:

```
28176486|16
86 20 8883 3888|5
+853 2833 7676|3
+852 31051830, +852 31041831, +852 31051832|2
+852 3481 1810|2
+853 2850 2525|2
+853 2853 3544|2
+853 2872 6288|2
+853 2878 2782|2
+853 2883 7788|2
```

The international country code for some numbers are absent, such as the numbers only with 8 digits. Some numbers started with “+86” or “86” indicate they may belong to areas in mainland China. In addition, there are a bunch of numbers started with “+853”, indicating they either belong to Macau, or they are mistakes.

To deal with the phone number inconsistency, I tried to put the phone numbers into regular expression, parsing them into groups. Then I programmatically corrected the problematic codes started with absent or “+853” in audit.py using the following function:

```
phone_pattern_re = re.compile(r'\D?(\d{0,4}?)\D{0,2}(\d{4})\D?(\d{4})
$', re.VERBOSE)

phone_mapping = { "", "853"}

def update_phone(phone, phone_mapping):
    results = []
    for iphone in re.split(',|;',phone):
        patterns = phone_pattern_re.search(iphone)
        if patterns:
            numbers = patterns.groups()
            if numbers[0] == "852":
                results.append(re.compile(r'\D?(\d{0,4}?)\D{0,2}
(\d{4})\D?(\d{4})$', iphone))
            elif numbers[0] in phone_mapping:
                results.append ("+852"+ " " + numbers[1] + numbers[2])
            return ';'.join(results)
```

This updated all substrings in problematic country codes, such that:

```
2196 8170 => +852 21968170
2736 0868 => +852 27360868
+853 2833 7927 => +852 28337927
+853 2856 8288 => +852 28568288
```

Data Overview and Additional Ideas

This section contains basic statistics about the dataset, the SQL queries used to gather them, and some additional ideas about the data in context.

File sizes

```
524381500 May 13 12:25 hong-kong_china.osm
433334272 May 16 15:17 hongkong.db
205059894 May 16 12:44 nodes.csv
6170640 May 16 12:44 nodes_tags.csv
15059326 May 16 12:54 ways.csv
70380086 May 16 12:54 ways_nodes.csv
23481938 May 16 12:54 ways_tags.csv
```

Number of nodes

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

2506176

Number of ways

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

256776

Number of unique users

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

1645

Top 10 contributing users

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

```
ORDER BY num DESC
LIMIT 10;
```

```
hlaw,501492
MarsmanRom,234505
Popolon,160601
sn0wblind,121198
fsxy,98243
katpatuka,96604
KX675,79276
fdulezi,79028
rainy3519446,58106
tomlee721,52607
```

Number of users only having 1 post

```
sqlite> SELECT COUNT(*)
FROM
    (SELECT e.user, COUNT(*) as num
     FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
     GROUP BY e.user
     HAVING num=1)  u;
```

```
295
```

Top 10 appearing amenities

```
sqlite> SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

```
restaurant,769
shelter,660
bus_station,567
bank,561
parking,557
fast_food,329
fuel,238
taxi,230
place_of_worship,221
```

Most popular cuisines

```
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 10;

chinese,137
japanese,21
indian,15
italian,9
pizza,9
regional,9
international,8
burger,7
thai,7
french,6
```

Top 10 banks

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='bank') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='name'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 10;

HSBC|35
工商银行|35
中国银行|24
中国工商银行|17
BEA|14
中国农业银行|14
中国建设银行|14
BOC|11
```

```
ICBC|11
Standard Chartered|11
```

Additional Ideas

Contributor statistics

Similar to other regions in the OpenStreetMap project, the contributions of users seems skewed, as expected. Here are some user percentage statistics:

- Top user contribution percentage (“hlaw”) 18.15%
- Combined Top 10 users contribution 53.63%

Amenity statistics

Restaurants was the top appearing of amenities in Hong Kong. If combining restaurants and fast_food, there were total 4132 counts, 26.57% of the top 10 appearing amenities. In addition, bank ranked the 4th in the amenities, indicating Hong Kong as an international cuisine and financial center. The top bank is HSBC, which is an abbreviation of The Hongkong and Shanghai Banking Corporation, the biggest local bank here.

Even though restaurants and banks are listed as top amenities, the number counts of them are still low. For example, only 137 Chinese restaurants are documented in this map. This indicates that most of places is far from completed, especially when turning to the detail information of amenities.

As an international food and financial center, many international cuisines can be found in Hong Kong, from street-food stands to fine-dining restaurants. One popular application called OpenRice obtains almost all the restaurants in Hong Kong, with customers’ ratings. It is an HongKong version of Yelp. I think this would be a source to enhance the restaurant details in the OpenStreetMap if we can incorporate these local data. However, OpenRice as a commercial group, fetching data from them will be a potential problem. Compared to many open-source webpages, local small companies usually more tend to protect their data. In addition, a more complex different database schema may be needed if this data will be incorporated. Restaurant information from OpenRice usually contains the following features: cuisine, location, price, recommended menu, customers’ reviews, promotions, online table booking and so on. Therefore, I will foresee that incorporating such massive data into the map database will be a challenge.

In summary, the OpenStreetMap project contains lots of open-source data, flexible enough for explorations quantitatively and qualitatively. If incorporating user reviews of establishments, all this information, such as the number of amenities and the distribution of banks, can be used as a source to evaluate the quality of an neighborhood. The data might help people make better decisions on choosing neighborhood, buying an property estate, choosing schools for kids and so on.

Conclusion

After this review of the data I noticed that the Hong Kong area is far from completed, though I believe it has been well cleaned on the street names for the purposes of this exercise. OpenStreetMap project is a very good start for data wrangling beginners like me. I have been making progress on learning basic codings for data wrangling, as well as using SQL to review the database.

OpenStreetMap is more than just a geographical map. Interesting information can be extracted by SQL queries. In this project, I took a quick search on the amenities of Hong Kong. I believe this will provide more information if it can be improved with more details.