Wrangle OpenStreetMap Data Using SQL

Bin Liu

Map Area : Shenzhen, Guangdong ,China

Dataset This extract actually contains information for the Shenzhen City. Over the past 15 years, I have worked and lived in this city. I would like to change the angle from the OSM map data to see the city.

1. Problems Encountered in the Map

After initially downloading a small sample size of the Shenzhen City and running it with a provisional data.py file, I found the three main problems in the dataset, which I will discuss in the following order:

- Overabbreviated street type names ("盐梅路 Yanmei Rd")
- Pinyin spelling of Chinese words ("南山大道 Nanshan Dadao")
- Inconsistent city name ("深圳市","广东省深圳市")

1.1 Overabbreviated Street Names

After searching the street key type "addr:street" on the sample osm file with grep command, I found many street name has overabbreviated.

```
grep command: grep -i "addr:street" sample.osm
searching result:
<tag k="addr:street" v="罗芳路 Luofang Rd" />
<tag k="addr:street" v="文锦中路 Wenjin Middle Rd" />
<tag k="addr:street" v="香梅路 Xiangmei Rd" />
......
```

I solved the overabbreviated problem by first use the regular expression to filter out the street type, Then using a mapping dictionary (street_abbrev_mapping in config.py) to map the abbreviated name to countpart full name.

```
%run audit.py shenzhen_sample.osm -t street -p overabbreviated
```

```
audit.py: auditing the tag street for overabbreviated problem ...
正云路 Zhengyun Rd => 正云路 Zhengyun Road
景田西路 Jingtian W Rd => 景田西路 Jingtian W Road
莲花路 Lianhua Rd => 莲花路 Lianhua Road
Houhaibin Rd => Houhaibin Road
香梅路 Xiangmei Rd => 香梅路 Xiangmei Road
Wanghai Rd => Wanghai Road
```

```
Zhenghua Rd => Zhenghua Road
Gongye 7th Rd => Gongye 7th Road
景田路 Jingtian Rd => 景田路 Jingtian Road
比亚迪路 Biyadi Rd => 比亚迪路 Biyadi Road
罗芳路 Luofang Rd => 罗芳路 Luofang Road
Guihua Rd => Guihua Road
文锦中路 Wenjin Middle Rd => 文锦中路 Wenjin Middle Road
正风路 Zhengfeng Rd => 正风路 Zhengfeng Road
沙湾路 Shawan Rd => 沙湾路 Shawan Road
景田北街 Jingtian N Rd => 景田北街 Jingtian N Road
湖贝路 Hubei Rd => 湖贝路 Hubei Road
延芳路 Yanfang Rd => 延芳路 Yanfang Road
粤兴三道 Yuexing 3rd Rd => 粤兴三道 Yuexing 3rd Road
Guanlan Longhua New District => Guanlan Longhua New DiStreet
Hei Yuen St => Hei Yuen Street
Yan Hing St => Yan Hing Street
景田东一街 Jingtian E 1st St => 景田东一街 Jingtian E 1st Street
景田北三街 Jingtian North 3rd St => 景田北三街 Jingtian North 3rd Street
景田北七街 Jingtian North 7th St => 景田北七街 Jingtian North 7th Street
Fu Shin St => Fu Shin Street
景田北一街 Jingtian North 1st St => 景田北一街 Jingtian North 1st Street
```

Benefits of improve and expectations of the problem

This updated all substrings in problematic address strings, such that: “比亚迪路 Biyadi Rd” becomes: “比亚迪路 Biyadi Road”.

1.2 Pinyin spelling of Chinese Words

Pinyin spelling of chinese words is an interesting case. Often times, the chinese pinyin of a word is written in the english street name. For example "南山大道 Nanshan Dadao" instead of "南山大道 Nanshan Avenue ", both street name "Nanshan" and street type "Dadao" are using pinyin word. (see http://wiki.openstreetmap.org/wiki/WikiProject_China#Generics_in_Chinese).

Because pinyin spelling street name is a common custom. So I not deal with street name and instead focus on fix pinyin spelling street type problem. I solved the problem by using a mapping dictionay (pinyin_mapping in config.py) to map pinyin word to countpart english names.

```
%run audit.py shenzhen_sample.osm -t street -p pinyin
```

```
audit.py: auditing the tag street for pinyin problem ...
Houhaibin Rd => 华强北路 Huangqiang North Road
沙河西路 ShaHe Xi Lu => 沙河西路 ShaHe West Road
KeYuan Lu => KeYuan Road
Hongbao Lu => Hongbao Road
桃园路 Taoyuan Lu => 桃园路 Taoyuan Road
Shenyan Lu => Shenyan Road
```

```
Gaoxin S. => 高新区南
南山大道 Nanshan Dadao => 南山大道 Nanshan Avenue
```

Benefits of improve and expectations of the problem

This update making the name of the street type more consistent and comply with international practice. The better solution is create a coding or data standards, very good implementation of data standards when user input data. The potential problem is the lack of flexibility, and the ability to combat users who are unfamiliar with the new system.

1.3 Inconsistent city name

City name (addr:city) is Inconsistent, sometime is chinese ("深圳市","广东省深圳市"), sometime is pinyin word (“Shenzhen""). I solved the problem by using a mapping dictionay to map pinyin word to a unified name("深圳 Shenzhen").

```
%run audit.py shenzhen_sample.osm -t city -p Inconsistentname
```

```
audit.py: auditing the tag city for Inconsistent name problem ...
深圳市 => 深圳 Shenzhen
深圳深圳 => 深圳 Shenzhen
深圳市南山区 Nanshan District => 深圳 Shenzhen
深圳市 Shenzhen => 深圳 Shenzhen
深圳市罗湖区 => 深圳 Shenzhen
深圳市宝安区 => 深圳 Shenzhen
shenzhen => 深圳 Shenzhen
广东省深圳市 => 深圳 Shenzhen
Shenzhen City => 深圳 Shenzhen
```

Benefits of improve and expectations of the problem

This update making the field of city name more consistent, which make the subsequent data analysis more convenient. But a potential problem is lost some information, for example "深圳市宝安区" become "深圳 Shenzhen". Optional workaround is increase a new key "(addr:district") and save the information "宝安区" with it.

2. Data Overview and Additional Ideas

File sizes

```
shenzhen_sample.osm 70 MB
shenzhen.db 37M
nodes.csv 28M
nodes_tags.csv 905K
ways.csv 2.2M
ways_nodes.cv 9.7M
ways_tags.csv 3.1M
```

Number of nodes

```
sqlite> select count(*) from Nodes;
340639
```

Number of ways

```
sqlite> select count(*) from nodesTags ;
47530
```

Number of unique users

```
sqlite> SELECT COUNT(DISTINCT(e.uid))
   ...> FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
698
```

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;
MarsmanRom,88223
hlaw,61685
HelioFelix,20206
"Philip C",12846
samho1234567,11019
a1579,10673
jc86035,7249
ch40s,7173
eversone,6765
ssheldonss,6487
```

The user's contribution presents a very serious tilt. A small number of users contribute the majority of map information. Here, there are some statistical information about user contributions:

- Top user contribution percentage ("MarsmanRom") 23.34%
- Combined top 2 users' contribution ("MarsmanRom" and "hlaw") 39.66%
- Combined Top 10 users contribution 61.45%
- Combined number of users making up only 1% of posts 681 (about 97.28% of all users)

Observing these user percentages, I'm reminded of "gamification" and "pay mechanism" as a motivating force for contribution like Foursquare. From the game theory, the use of a mechanism called doubled or zero. Use this pay mechanism, Users can rate venues by answering questions, then give a reward for a valid answer. This can effectively reduce the data mark error rate, improve data quality.

But the potential problem is "gamification" cannot drive long-term behavioral change. the activity or behavior promoted needs to have an intrinsic value. At first, people were going out of their way to earn OpenStreetMap badges and bounty. But these rewards didn’t suffice in the longer term because once novelty wears off and the fun becomes yesterday’ s news, users need additional value. just like "开心网"(www.kaixin001.com).

3. Additional Data Exploration

3.1 Observe Tags

Connect to the osm database, Run a query to get type and number of occurrences for all tags

```
import sqlite3

db = sqlite3.connect("shenzhen.db")
c = db.cursor()
QUERY = '''
select key, count(*) as count
from (select * from nodesTags union all select * from waysTags) tags
where key is not Null
group by key
order by count desc;
'''
c.execute(QUERY)
rows = c.fetchall()
import pandas as pd
df = pd.DataFrame(rows)
db.close()
```

Frequency of Tag

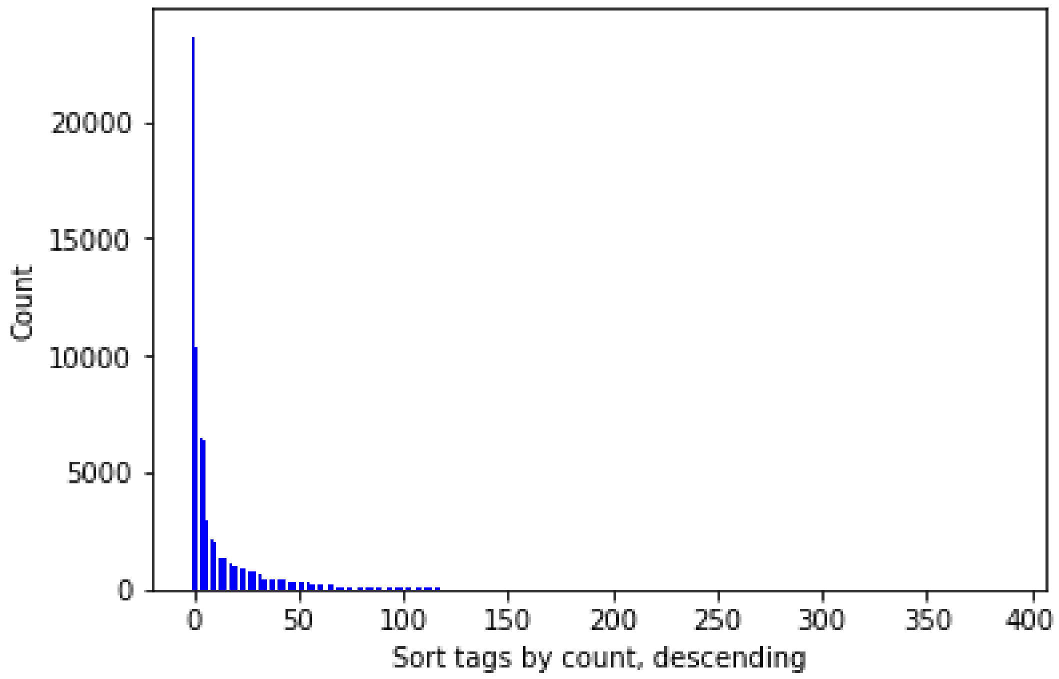
```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

tag_ranks = df[0]
tag_counts= df[1]

N = len(tag_counts)
tag_ranks = range(N)
width = 1/1.5

plt.xlabel("Sort tags by count, descending")
plt.ylabel("Count")
plt.bar(tag_ranks, tag_counts, width, color="blue")
```

```
<Container object of 388 artists>
```



Top 10 tags with the highest frequency

```
df.columns = ['key of tags' ,'count']
print df[0:10]
```

```
  key of tags  count
0     highway  23632
1        name  10358
2    building   7665
3          en   6953
4      oneway   6482
5          zh   6329
6       power   2961
7       layer   2879
8      bridge   2398
9      source   2104
```

3.2 Amenities By Count

```
sqlite> SELECT value, COUNT(*) AS num
   ...> FROM nodesTags
   ...> WHERE key = 'amenity'
   ...> GROUP BY value
   ...> ORDER BY num DESC;
```

```
toilets,198
restaurant,133
shelter,100
post_box,87
bank,69
```

```
bus_station,56
fuel,50
fast_food,48
bicycle_parking,42
cafe,39
```

3.3 Most Popular Cuisines

```
sqlite> SELECT value, COUNT(*) AS num
   ...> FROM nodesTags
   ...> WHERE key = 'cuisine'
   ...> GROUP BY value
   ...> ORDER BY num DESC;
chinese,31
burger,12
chinese;fast_food,4
japanese,4
chicken,3
regional,3
vegetarian,3
cantonese,2
local,2
mexican,2
noodle,2
Dumplings,1
coffee_shop,1
```

4. Conclusion

This is a very challenging project, data cleaning is far from enough. I only focused on auditing and cleaning the addr:street and addr:city tags. For example name and building fields contains many records from other city other than the expected '深圳 Shenzhen'. The audit and fix functions are not applied to clean the data before writing into csv files.

I can imagine the challenge in using these approaches to other tags. If the user generated data have standards to follow and auditing tools to improve accuracy and consistancy, the OSM data could be put into broad application.