OpenStreetMap Data Case Study

Map Area

Nanjing , Jiangsu , China

- https://www.openstreetmap.org/export#map=8/31.520/119.647
- http://overpass-api.de/query_form.html

(using querying code: (node(31.551,117.960,32.621,119.647);<;);out meta;)

In fact, the initially downloaded XML file maybe contains a part of data about Nanjing,and the other information is relative to some other cities. My university is in Nanjing , the capital city of Jiangsu Province.I am likely to stay here for development after graduation,so I am more interested to see what message I can get from database querying.

Problems Encountered in the Map

Creating a provisional test.py file,I noticed four main problems with the data,which I will discuss in the following order:

- Incorrect street names(“珠江路 8 号”,” 宏运大道 1188 号”)
- Incomplete postal codes(“21351”)
- Different key identifiers and values about Wikipedia tags:
  <tag k="wikipedia:zh" v="崔致远纪念馆"/>
  <tag k="wikipedia" v="zh:长深高速公路"/>
- Nonstandard city names(“南京市雨花台区”,”镇江市句容市”,”碧桂园凤凰城大凤凰”)

Incorrect street names

When a number is added after a street name, an incorrect street name appears. I collect the street names by using the following codes:

```
addr_colon_div = re.compile(r'addr:(.*)')
match = addr_colon_div.search( element.attrib["k"])
if match:
  if  (match.group(1) =="street"):
     print (element.attrib["v"])
```

I have specified the key(“street”) and the type(“addr”) when I filter the tags, so I am pretty sure the street name with the number is wrong.To correct them, I use the following function in audit.py :

```
def update_streetName(name):
   zhPattern = re.compile(u'[\u4e00-\u9fa5]+')
   match = zhPattern.search(name)
   if match:
     return (match.group(0))
   else:
      return (name)
```

This updated all this type of incorrect street names,such that:
“珠江路 8 号” becomes: “珠江路”
”宏运大道 1188 号” becomes: “宏运大道”

Incomplete postal codes

I use the following codes to find out all the postal codes in my OSM file:

```
addr_colon_div = re.compile(r'addr:(.*)')
match = addr_colon_div.search( element.attrib["k"])
if match:
    if  (match.group(1) =="postcode"):
        print (element.attrib["v"])
```

I glanced that one of them(the output postal codes) is different from the others,which is only five digits while others are all six. Postal codes in our country are usually six figures,so I decided to search the original OSM file for the tag for just one such fault. I can search the Internet according to Longitude and latitude information from its parent and fix it finally .
Regardless,after fixing incomplete postal codes,some altogether postal codes surfaced when grouped together with this aggregator:

```
select tags.value , count(*) as count
from (select * from nodes_tags
union all
```

```
select * from ways_tags)  as 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
212003|48
210000|9
210023|7
210008|6
210002|5
210005|3
210036|3
210093|3
211100|3
210013|2

After consulting the relevant documents of the post office,I found all postal codes land in the chooesd area. 212003 belongs to the postcode of Zhenjiang city,which is very close to Nanjing,and others for Nanjing city. Particularly, 211100 is exactly pointed to the district my campus locates.So excited!

Nonstandard city names

```
addr_colon_div = re.compile(r'addr:(.*)')
match = addr_colon_div.search( element.attrib["k"])
if match:
   if  (match.group(1) =="city"):
       print (element.attrib["v"])
```

The above codes is for me to check whether the name of a city is correct. However,three major abnormal situations occurred :

1. A district belonging to a city,like this (“南京市雨花台区”)

Yuhuatai should belong to a region of Nanjing, and cannot be labeled as a city.

2. Superposition of two city names,like this(“镇江市句容市”)

It is likely to be a boundary node in the map,but it is also an error city name.

3. A particular location Too Specific to be a city name,like this(“碧桂园凤凰城大凤凰”) For this case, I decided to change ite key value.

About Item2 ,the node’s geographic information maters.It need change manully.

About Item1,I coded a function to process it automatically. The source is below:

```
def update_cityName(name):
    changeList  = ["江苏省南京","江苏省南京市","南京","南京江宁区","Nanjing","NANJING","nanjing"]
    Pattern = re.compile(u'(.*)市(.*)区')
    match = Pattern.search(name)
    if match:
       return (match.group(1)+"市")
    else:
       if name in changeList:
          name = "南京市"
          return name
       return name
```

After function processing , “南京市雨花台区” becomes “南京市”,which is Nanjing city in English.

Different key identifiers and values about Wikipedia tags

When I checked the key-value data in tags,I found tags related to Wikipedia explaination have different formats. In order to establish database and query,I should change one and save another for convenience. I think tags like this : <tag k="wikipedia:zh" v="崔致远纪念馆"/> are of higher readability.

```
wikipedia = re.compile(r'(.*)wikipedia(.*)')
match = wikipedia.search( element.attrib["k"])
if match:
   print (element.attrib["k"],"&",element.attrib["v"])
```

The above codes is for my inspection. Thus I have written a function which can update the values to a preferable format.The function source code are displaying below.

```
def update_WikipediaValue(name):
   Pattern = re.compile(r'(.*):(.*)')
   match = Pattern.search(name)
   if match:
     return (match.group(2))
   else:
      return (name)
```

Sort cities by count ,descending

```
select tags.value , count(*) as count
from (select * from nodes_tags
union all
select * from ways_tags)  as tags
where tags.key like '%city'
group by tags.value
order by count desc;
```

And,the result,edited for readability:

"南京市",93
"扬州",4
50,3
"镇江",3
100,2
24,2
32,2
10,1
13,1
14,1
15,1
"2 GW",1
"2400 MW",1
27,1
29,1
30+,1
31,1
33,1
"3980 MW",1
40,1
"435 MW",1
52,1
6,1
"640 MW",1
"760 MW",1
"780 MW",1
8,1
"溧阳市",1

These result reveals that most values in the selected area are relative to Nanjing city,which is excepted.“扬州”and” 镇江”are both cities around here.
When I inspected the resulte list, some values looks different from others,for example, the number(or perhaps with few letters) values. This makes analysis difficult exactly. So my advice is contributors should perfectly clear the type of every value and normalize own definition before committing.

Now I am interested in the value” 2 GW. Let’s display part of its document for closer inspection(for our purposes,only limited fields are relevant):

```
select *
from ways where id in (
select distinct(id) from ways_tags where value = '2 GW' );
```

The data returned from the query is below.
168370572|Olyon|1443767|2|43113866|2016-10-24T04:09:40Z

And then:

```
select * from ways_tags where id = '168370572';
```

168370572|method|combustion|generator
168370572|source|coal|generator
168370572|landuse|industrial|regular
168370572|name|句容电厂|regular
168370572|en|Jurong Power Plant|name
168370572|ja|句容発電所|name
168370572|output:electricity|2 GW|plant
168370572|power|plant|regular
168370572|source|wikipedia|regular
168370572|en|List of major power stations in Jiangsu province|wikipedia

It turns out,in the section sorting the cities I queried by using “like ‘%city’”,so this value was grabbed.Now I know it is a power plant in Jurong city. And the value “2 GW” is used to describe its power generation. Maybe I need make my constraints more specific next time I query the database.

Data Overview and Additional Ideas

This section contains basic statistics about the dataset, and some additional ideas about the ideas about the data in context.

File sizes

OSM.sqlite.....95.4MB
OSM.osm 134MB
nodes.csv.....52.4MB
nodes_tags.csv.....843KB
ways.csv.....4.61MB
ways_nodes.csv.....18.8MB
ways_tags.csv.....4.62MB

Number of nodes

```
select count(*) from nodes;
```

652194

Number of ways

```
select count(*) from ways;
```

79688

Number of unique users

```
select count(distinct(sub.uid)) as num
 from (select uid from nodes
 union all
select uid from ways) as sub;
```

578

Top 15 contributing users

```
select sub.user,count(*) as num
 from (select user from nodes
 union all
select user from ways) as sub
group by sub.user
order by num desc
limit 15;
```

Chen Jia|292708
四国军棋军长|43098
bhw98|35225
sinopitt|35054
ff5722|31728
katpatuka|23590
jerryhappy|20859
市川ゆり子|15779
Sunng|15172
Metrojuze|12264
jamesks|12264
Metro Juze|12154
babribeiro|8720
FreedSky|7642
lorkhan|6898

Number of users appearing only once(having 1 post)

```
select count(*)
from (select e.user,count(*) as num
         from (select user from nodes
          union all
        select user from ways) as e
         group by e.user
         having num = 1 ) as u;
```

87

Additional Ideas

What a data Analyst can search from the dataset is not a little, but I think it is not the final purpose. Although cleaning the data is an indispensable work for them, the normative standards is necessary.The organization OpenStreetMap.org should formulate and execute these standards. And in the other hand,encouraging the contributors to observe the rules at their every commit will make a difference. Guide for the standards is needed and must be easily accessible. No body wants his achievements look messy,so this can convey a message : what you do is helpful for others not trouble.
In fact,the benefit from the standard data collection procedure is obvious.For example,an analyst needing some specific dataset can code a function to index the database from OpenStreetMap efficiently guided by data storage specifications. Confusing storage will only make people stand at arm's length. Huge amounts of data will benefit from orderly placement.

Additional Data Exploration

Top 5 appearing amenities

```
select value,count(*) as num
from nodes_tags
where key='amenity'
group by value
order by num desc
limit 5;
```

restaurant|191
bank|156
bicycle_rental|151
toilets|85
fast_food|72

Nowadays, the sharing economy is booming in my country, so I'm not surprised that bike rental rankings are so high.

What is most popular landuse

```
select nodes_tags.value,count(*) as num
from nodes_tags
join (select distinct(id) from nodes_tags where value = 'residential') as IDset
on nodes_tags.id= IDset.id
group by nodes_tags.value
order by num desc;
```

residential|27
南苑真园|2
ZhenYuan|1
长江之家|1
长阳花园|1
东宝花园|1
桂苑|1
荷苑|1
江苏省老年公寓|1
金陵世纪花园|1
菊苑|1
兰苑|1
李苑|1
柳苑|1
梅苑|1
七彩星城|1
七彩星城-国学府|1
七彩星城-幸福里|1
融侨世家(东门)|1
融侨世家(西门)|1
苏宁馨瑰园|1
桃苑|1
西泉|1
银城花园|1
竹苑|1

Conclusion

After this review of the data it is obvious that the Nanjing area is remarkably incomplete,though I believe it has been well cleaned for the purposes of this execise.I have accumulated a lot of interest for the next learing about data analysis.

With professional software and scientific methods,I think the important infomation embedded in huge data will be presented to me.I can not wait !