

数据来源:

<https://www.openstreetmap.org/relation/3987435>

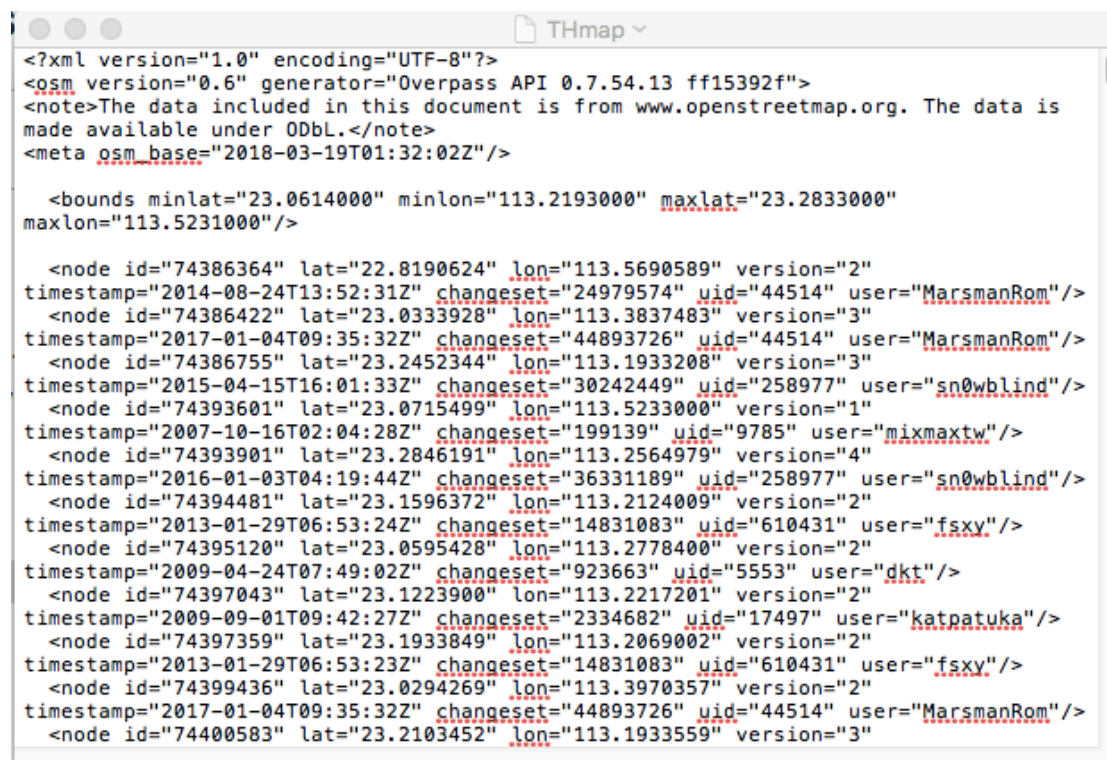
我选择了广东省广州市天河区，选择这个区域原因有以下：

1. 我住在广州，对天河有比较深入的了解
2. 我其实刚开始选择广州的，但是由于太大以及刚开始不会下载的原因失败了
3. 其次尝试天河区的地图结果成功了。而且天河区的数据 82.3MB，也符合要求，便选择了这个。

数据审查以及导出过程

1. 由于这个数据是 82.3MB，对于地图的数据来说这个大小不算大，便直接拿来运行代码。

数据是这个样子的：



```
<?xml version="1.0" encoding="UTF-8"?>
<qsm version="0.6" generator="Overpass API 0.7.54.13 ff15392f">
<note>The data included in this document is from www.openstreetmap.org. The data is
made available under ODbL.</note>
<meta osm_base="2018-03-19T01:32:02Z"/>

<bounds minlat="23.0614000" minlon="113.2193000" maxlat="23.2833000"
maxlon="113.5231000"/>

<node id="74386364" lat="22.8190624" lon="113.5690589" version="2"
timestamp="2014-08-24T13:52:31Z" changeset="24979574" uid="44514" user="MarsmanRom"/>
<node id="74386422" lat="23.0333928" lon="113.3837483" version="3"
timestamp="2017-01-04T09:35:32Z" changeset="44893726" uid="44514" user="MarsmanRom"/>
<node id="74386755" lat="23.2452344" lon="113.1933208" version="3"
timestamp="2015-04-15T16:01:33Z" changeset="30242449" uid="258977" user="sn0wblind"/>
<node id="74393601" lat="23.0715499" lon="113.5233000" version="1"
timestamp="2007-10-16T02:04:28Z" changeset="199139" uid="9785" user="mixmaxtw"/>
<node id="74393901" lat="23.2846191" lon="113.2564979" version="4"
timestamp="2016-01-03T04:19:44Z" changeset="36331189" uid="258977" user="sn0wblind"/>
<node id="74394481" lat="23.1596372" lon="113.2124009" version="2"
timestamp="2013-01-29T06:53:24Z" changeset="14831083" uid="610431" user="fsxy"/>
<node id="74395120" lat="23.0595428" lon="113.2778400" version="2"
timestamp="2009-04-24T07:49:02Z" changeset="923663" uid="5553" user="dkt"/>
<node id="74397043" lat="23.1223900" lon="113.2217201" version="2"
timestamp="2009-09-01T09:42:27Z" changeset="2334682" uid="17497" user="katpatuka"/>
<node id="74397359" lat="23.1933849" lon="113.2069002" version="2"
timestamp="2013-01-29T06:53:23Z" changeset="14831083" uid="610431" user="fsxy"/>
<node id="74399436" lat="23.0294269" lon="113.3970357" version="2"
timestamp="2017-01-04T09:35:32Z" changeset="44893726" uid="44514" user="MarsmanRom"/>
<node id="74400583" lat="23.2103452" lon="113.1933559" version="3"
```

2. 审查数据

由于原始数据一眼看上去得不到什么有效的信息，因此用了代码来获取我希望得到的信息。

我用了如下的方法来审查数据。（代码具体在 DAP3_1_audit_clean）

```
def count_tags(filename):
    count = defaultdict(int)
    for node in ET.iterparse(filename):
        # print node[1].tag
        count[node[1].tag] += 1

    return count
```

```
defaultdict(<type 'int'>, {'node': 340581, 'member': 121970, 'nd': 392909, 'tag': 137940, 'bounds': 1, 'note': 1, 'meta': 1, 'relation': 1790, 'way': 41245, 'osm': 1})
```

node 有 340581 条，'member' 有 121970，'nd' 有 392909，'tag' 有 137940，'bounds' 有 1，'note' 有 1，'meta' 有 1，'relation' 有 1790，'way' 有 41245，'osm' 有 1，从这个结果大概可以理清楚这个 map 数据的结构以及多少。

接下来检查 k 值是否存在问题

```
lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\/&<>\'\"?%#$@\\,\\. \t\r\n]')

def key_type(element, keys):
    if element.tag == "tag":

        if lower.search(element.attrib['k']) != None:
            keys['lower'] += 1

        elif lower_colon.search(element.attrib['k']) != None:
            keys['lower_colon'] += 1

        elif problemchars.search(element.attrib['k']) != None:
            keys['problemchars'] += 1
        else:
            keys['other'] += 1
    return keys

def process_map(filename):
    keys = {"lower": 0, "lower_colon": 0, "problemchars": 0, "other": 0}
    for _, element in ET.iterparse(filename):
        keys = key_type(element, keys)
    return keys
```

“lower”，表示仅包含小写字母且有效的标记，

“lower_colon”，表示名称中有冒号的其他有效标记，

“problemchars”，表示字符存在问题的标记，以及

“other”，表示不属于上述三大类别的其他标记。

```
{'problemchars': 0, 'lower': 120152, 'other': 137, 'lower_colon': 17651}
```

说明该文件不存在问题的标记，存在冒号的有效标记，意味着后面处理的时候这部分需要处理。

接下来具体审查接到名：

用以下方法：

```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
street_types = defaultdict(set)

expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Ct", "Place", "Square", "Lane", "Road",
            "Trail", "Parkway", "Commons", "Park", "Broadway", "Circle", "Highway", "Trail",
            "Way", "West", "North", "Terrace", "Plaza", "Market"]
mapping = { "St": "Street", "St.": "Street", "ST": "Street", "street": "Street", "st": "Street", 'Jie': 'Street',
            'jie': 'Street',
            "Rd": "Road", "raod": "Road", "road": "Road", "Lu": "Road",
            "Ln": "Lane",
            "BLVD": "Boulevard",
            "Acenue": "Avenue", "Ave": "Avenue", "avenue": "Avenue", "Av": "Avenue",
            "Hwy": "Highway",
            "Blvd": "Boulevard",
            "Ct": "Court",
            "E": "East", "S": "South", "W": "West", "N": "North", "S.": "South",
            "NE": "Northeast", "NW": "Northwest", "SE": "Southeast", "SW": "Southwest",
            "Dadao": "DaDao"
            }

def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)
```

```
def is_street_name(elem):
    return elem.attrib['k'] == "addr:street"
# return elem.attrib['k'] == "name:"

def audit(osm_file):
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == "way" or elem.tag == "node":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    pprint.pprint(dict(street_types))
    osm_file.close()
    return street_types
```

输出部分结果：

```
'Guangzhou': set(['Room 1302, E2 Building, Jin Gui Yuan, #6 Jin Gui Jie, Jie Fang Bei Lu, Bai Yun District, Guangzho
u']),
'u'Lu': set(['u\u6ee8\u6c5f\u4e1c\u8def Binjiang Dong Lu']),
'u'N': set(['u\u5e7f\u5dde\u5927\u9053\u5317 Guangzhou Ave N']),
'u'Rd': set(['Linhe West Cross Rd',
            'Liurong Rd',
            'u\u5185\u73af\u8def Inner Ring Rd']),
'u'Shop': set(['u\u6c99\u592a\u5357\u8def, upstairs in Health food Shop']),
'St': set(['Yanyu S St']),
'Xi': set(['Huang Pu Dadao Xi']),
'ave': set(['baogang ave']),
'road': set(['Huacheng road']),
'road': set(['Chigang Lu (road)'])
defaultdict(<type 'set'>, {'u'Shop': set(['u\u6c99\u592a\u5357\u8def, upstairs in Health food Shop']), 'Guangdong': s
et(['Room 322, Jintao Building, 26# Guangyuanzhong Road, Baiyun District, Guangzhou, Guangdong, ']), 'road': set(['Chig
```

可以明显的看出有些名字运用了缩写，例如

```

u'Rd': set(['Linhe West Cross Rd',
            'Liurong Rd',
            u'\u5185\u73af\u8def Inner Ring Rd']),
u'Shop': set([u'\u6c99\u592a\u5357\u8def, upstairs in Health food
Shop']),
'St': set(['Yanyu S St']),
'Xi': set(['Huang Pu Dadao Xi']),
'ave': set(['baogang ave']),

```

还出现了 Unicode 字符，因此需要中转换成拼音

例如

广州大道北 Guangzhou Ave N

```
(u'\u5e7f\u5dde\u5927\u9053\u5317 Guangzhou Ave N')
```

因此需要更新

更新名字:

```

def update_name(name, mapping):
    # for key in mapping:
    #     keyy=key
    ##     a=name.find(keyy)
    #     if a> 0 :
    ##         b=key
    #         b =mapping[b]
    #         name =name[:a]+b
    name = name.split()
    for i in range(len(name)):
        if name[i] in mapping:
            name[i] = mapping[name[i]]
    name = " ".join(name) #http://blog.csdn.net/chixujohnny/article/details/53301995
    return name

```

中文转英文

```

#识别是不是中文
def Chinese_English(name): #https://zhidao.baidu.com/question/1958984257226168500.html
    match = zhPattern.search(name)
    if match:
        return ''.join(lazy_pinyin(name))
    else:
        return name

osm_file=open("/Users/chebyshev/Downloads/THmap", "r")
filename="/Users/chebyshev/Downloads/THmap"

```

部分结果为:

```

广州市荔湾区西村街道办事处彭城东路58—60
(u'\u5e7f\u5dde\u5e02\u8354\u6e7e\u533a\u897f\u6751\u8857\u9053\u529e\u4e8b\u5f6d\u57ce\u4e1c\u8def58\u4e0060', '=>',
u'\u5e7f\u5dde\u5e02\u8354\u6e7e\u533a\u897f\u6751\u8857\u9053\u529e\u4e8b\u5f6d\u57ce\u4e1c\u8def58\u4e0060')
Yanyu S St
('Yanyu S St', '=>', 'Yanyu South Street')
内环路 Inner Ring Rd
(u'\u5185\u73af\u8def Inner Ring Rd', '=>', u'\u5185\u73af\u8def Inner Ring Road')
Linhe West Cross Rd
('Linhe West Cross Rd', '=>', 'Linhe West Cross Road')
Liurong Rd
('Liurong Rd', '=>', 'Liurong Road')
Lang wang Bilu
('Lang wang Bilu', '=>', 'Lang wang Bilu')
滨江东路 Binjiang Dong Lu
(u'\u6ee8\u6c5f\u4e1c\u8def Binjiang Dong Lu', '=>', u'\u6ee8\u6c5f\u4e1c\u8def Binjiang Dong Road')
广州市越秀区北站路168号
(u'\u5e7f\u5dde\u5e02\u8d8a\u79c0\u533a\u5317\u7ad9\u8def168\u53f7', '=>', u'\u5e7f\u5dde\u5e02\u8d8a\u79c0\u533a\u5317\u7ad9\u8def168\u53f7')
港前路531号大院
(u'\u6e2f\u524d\u8def531\u53f7\u5927\u9662', '=>', u'\u6e2f\u524d\u8def531\u53f7\u5927\u9662')
baogang ave
('baogang ave', '=>', 'baogang ave')
珠江东路 Zhujiang Road East
(u'\u73e0\u6c5f\u4e1c\u8def Zhujiang Road East', '=>', u'\u73e0\u6c5f\u4e1c\u8def Zhujiang Road East')
China, Guangdong Sheng, Guangzhou Shi, Tianhe Qu, TianHe GongYuan, Tianhe Rd, 太古汇L307号, 邮政编码: 510610
(u'China, Guangdong Sheng, Guangzhou Shi, Tianhe Qu, TianHe GongYuan, Tianhe Rd, \u592a\u53e4\u6c47L307\u53f7, \u90ae\u53f7\u7f16\u7801: 510610', '=>', u'China, Guangdong Sheng, Guangzhou Shi, Tianhe Qu, TianHe GongYuan, Tianhe Rd, \u592a\u53e4\u6c47L307\u53f7, \u90ae\u53f7\u7f16\u7801: 510610')
Huacheng road
('Huacheng road', '=>', 'Huacheng Road')

```

审查完之后，准备写入 csv 文件中：

其中 shape_element 函数，这个函数会将每个元素转换为正确的格式

node 会输出 return {'node': node_attribs, 'node_tags': tags}

way 会输出 return {'way': way_attribs, 'way_nodes': way_nodes, 'way_tags': tags}

准备了 5 个 csv 文件，分别是“nodes.csv”、“nodes_tags.csv”、“ways.csv”、“ways_nodes.csv”、“ways_tags.csv”

导入 sql 后，用了基本的 sql 语句观察这个数据集：

首先，检查该数据集是否正确

```

[sqlite> select count(*) from nodes;
340582

```

正确应该是 340581，但是由于数据集第一行是标题，所以无误。

```

sqlite> select count(*) from ways
...> ;
41246

```

正确应该是 41245，同样因为数据集首行是标题，所以无误。

```

select user, count(*) as num from ways group by user order by num desc
limit 3;

```

看看 ways 里面最活跃的前三人是谁

```
[sqlite> select user, count(*) as num from ways group by user order by num desc \l  
limit 3;  
yangjiaozhongshihei|14059  
fdulezi|6599  
sn0wblind|4207  
sqlite> ]
```

看看麦当劳，kfc，必胜客总共有多少家！？

```
select value, count(*) from nodes_tags where (value="Pizza Hut" or  
value="KFC" or value="McDonalds");
```

```
KFC|68 Pizza Hut|16 McDonalds|4
```

总结

过程中出现的问题：

1. 出现乱码

解决：是中文的英文，先识别是否中文，如果是，则用函数转成英文再更新名字

2. 输出 way 那三个 csv 时为空白

解决：是赋值错误，我在 ways 那部分的代码用了跟 nodes 那部分的代码一致，可能 ways 的数据跑去 nodes 表格去了，改了名字后，正常！

sql 的创建失败

解决：我刚开始是直接创建了表，然后导入数据，结果做完了发现找不到我的数据，应该是先创建 database，再创建表。而且由于对 sql 不熟悉，百度了好久的教程已经询问朋友，终于成功！

sql 输出为空白

仔细看了一下，并不是数据的错误，而是导入的时候并没有分割好！

id	key	value	type
id, key, value, type			
244080443,	capital		
244080443,	ADM1,	30	
244080443,	DSG,	ADM	
244080443,	UFI,	-19	
244080443,	UNI,	672	
244080443,	contine		
244080443,	country		
244080443,	country		
244080443,	iso_316		
244080443,	name,	gu	
244080443,	ca,	Cant	
244080443,	cs,	Kuan	
244080443,	da,	Guan	

后来在 import 数据前加了 .separator ,就好了! 真是非常感谢同学的帮助!

对数据的问题:

1. 对数据进行 sql 探索的时候发现该区域的数据不足, 或者说不正确。
据我所知麦当劳的数量绝对是大于 4 家的, 而这里只显示了 4 家。

建议:

1. 我觉得网站应该审核一下用户所修改或者提供的数据, 这样数据才能准确或者说齐全。
2. 在一些输入框固定住输入的类型, 例如电话这部分, 可以固定用户输入时必须按照规定的电话格式输入; 在输入类型时可以提供选项而非手工输入, 选项可以有, 餐厅, 医院, 公园, 商店……从而减少因为手工以及主观的因素而引起的错误。

好处:

- 1, 使用数据的工作者或者研究者能对数据更加的放心, 减少压力。
- 2, 更多的数据使用者愿意到该网站下载数据。

预期的问题:

1. 网站的成本会提高, 因为不仅需要请熟悉各个地域的人核查数据, 还需要请相关技术人员来调整页面的输入。

