OpenStreetMap 案例研究：武汉市数据

选择区域

中国·湖北省·武汉市

- www.openstreetmap.org/relation/3076268

选择该区域的理由：

1. 地处国内，方便信息检查
2. 数据集大小比较适宜

地图中存在的问题

通过对数据的样本进行初步的探索，包括直接打开文本文件查看以及载入数据库执行SQL语句等方式，发现了以下三个问题。

- 地点的中文名称存在繁简混杂的情况
- 存在一些地点名称不以中文显示
- 门牌号码显示不规范

统一中文名称

根据对该数据集中有关tags的初步分析，在地点的命名方面重点关注两类：

- 'key' == 'zh' and 'type' == 'name'
- 'key' == 'name' and 'type' == 'regular'

这两类的值都表示地点的名称，并且大部分都以简体中文形式显示。但在初步探索的过程中，我发现其中存在少量繁体中文名称，这不仅不符合大陆通行简体字的规范，对分组、搜索等功能也会造成负面影响（地名关键词不一致），因此我决定把数据集中ways_tags 和 nodes_tags中上述两类名称修改为简体中文，核心代码如下：

```
with open(filename, 'rb') as f:
    reader = csv.DictReader(f)
    reader.next()
    for row in reader:
        if row['type'] == 'name' and row['key'] == 'zh' or row['key'] == 'name':
            row['value'] = HanziConv.toSimplified(row['value'])
        row_list.append(row)
with open(filename, 'wb') as f:
    writer = csv.DictWriter(f, delimiter = ',', fieldnames = ['id', 'key', 'type', 'value'])
    writer.writeheader()
    writer.writerows(row_list)
```

这里使用了hanziconv扩展库，可以方便地进行中文的繁简转换。这样，写入的新CSV文件中所有的名称都会转化为简体中文。

将英文名称转换为中文

这一操作和繁简转换类似，关注的也是前述的两类本应该只有简体中文显示的名称，只是将注意点放在了语言的不同上。我发现在地点的名称上还存在少量英文名称，会给不熟悉英文的人带来困扰（英文名的规范应当为'key' == 'en' and 'type' == 'name'），因此在自己有限的知识范围内将其转换成了中文。由于自己并非武汉人，译名肯定有不不准确的地方，但方向上大体是正确的。

在转换过程中，我采用的还是比较原始的方法，即手工翻译 + 建立字典，如：

```
dic = {"McDonald's":'麦当劳','Bank of China':'中国银行'}
```

效率比较低，但还没找到更好的方法，因为即使是英文名称，不规范之处仍然很多，用现成的翻译库无法做到有效翻译（比如在nodes_tags中，“麦当劳”就存在'McDonald's''McDonald`s''McDonald‘s'三种写法）。

另外，要检查UNICODE字符串中是否含有中文，可以采用我搜索到的如下方法，也是清理英文名称的核心所在：

```
def is_chinese(s):
for c in s:
#中文字符的编码范围
    if c >= u'\u4e00' and c <= u'\u9fa5':
        return True
return False
```

值得注意的是，在名称处理过程中我还发现了诸如'123'或者'sdfsdf'等无意义内容。由于个人了解的有限，对这部分没有进行处理，但直接删除可能是一个可行办法。

统一门牌号格式

根据数据的初步探索，门牌号格式应当为纯数字，因此具体思路为：提取门牌号中的数字作为门牌号的值。但在具体操作过程中，可能会提取到一个以上的数字，例如“87号速8”（该名称不真实存在）。此情况下程序无法判断到底“8”还是“87”为正确的门牌号，因此需要手工处理。核心代码如下：

```
for row in reader:
    if row['key'] == 'housenumber':
        #正则表达式：提取字符串中的所有数字
        possible_numbers = re.findall(r'[0-9]+', row['value'])
        #如果只提取了一个，说明该数字是正确的门牌号，否则不能确定，需要手动修改。
        if len(possible_numbers) == 1:
            row['value'] = possible_numbers[0]
```

检查邮政编码格式

在数据探索过程中，邮政编码的格式大体上是正确的，但检查邮政编码作为数据审查的一个步骤是非常值得实施的。

由于各地名所对应的邮政编码十分复杂，因此该步骤不确保邮政编码的正确性，仅检查是否满足下列要求：

- 格式是否为六位纯数字；
- 前两位数是否为“43”（武汉市邮政编码的前两位数）。

对于不满足要求的，将其加入一个列表，以备审查。检查代码如下：

```
to_edit = []
def check_postcode(code):
    return code.isdigit() and len(code) == 6 and code.startswith('43')
for row in reader:
    if row['key'] == 'postcode':
        if not check_postcode(row['value']):
            to_edit.append(row)
```

数据概览

这一部分涵盖了数据集的基本信息和初步统计。

文件大小

```
wuhan.osm ........ 58.9 MB
wuhan.db ......... 31.5 MB
nodes.csv ........ 23.5 MB
nodes_tags.csv ... 0.45 MB
ways.csv ......... 1.8 MB
ways_tags.csv .... 2.0 MB
ways_nodes.csv ... 0.86 MB
```

节点数量

```
sqlite> select count(*) from nodes
```

293192

途径数量

```
sqlite> select count(*) from ways
```

30825

唯一用户的数量

```
sqlite> select count(distinct(u.uid))
from (select uid from nodes union all select uid from ways) u;
```

416

贡献最多的10个用户

```
sqlite> select user, count(user) as num
  from (select user from nodes union all select user from ways)
  group by user
  order by num desc
  limit 10
```

```
GeoSUN: 119536
Soub: 50107
Gao xioix: 18631
katpatuka: 17611
jamesks: 14882
dword1511: 14177
hanchao: 5475
zueler: 4769
liuzhebing: 4333
nuklearerWintersturm: 4263
```

出现最多的10类节点

```
sqlite> select key, count(*) as num
  from (select key from nodes_tags union all
     select key from ways_tags)
  group by key
  order by num desc
  limit 10
```

```
highway,17556
name,9041
oneway,7052
building,6965
power,3866
source,3432
lanes,2168
layer,2135
bridge,2123
natural,1947
```

可以看出，节点的类型并不规范。

其他想法

显然，该数据集是相当不完整的，清理工作也仅仅完成了一小部分，但基本上还是达到了项目的目的。我认为该数据集还能够做以下改进：

- 增加缺失数据。事实上该数据集的缺失数据是非常多的，例如，我们可以查询武汉市“华莱士”餐厅的数量（由于数据已经清理过，名称都是用中文标示

的；由于中文字符处理问题，该命令使用python的DB-API执行）：

```
c.execute('select value, count(*) from nodes_tags where key = "name" and value = "华莱士"')
o = c.fetchall()
for t in o:
  print t[0], t[1]
```

```
华莱士 3
```

通过查询，发现武汉市仅有3家“华莱士”餐厅，但通过百度地图查询，武汉市至少有20家以上华莱士餐厅，因此可以判断数据集中存在大量缺失。该改进带来的益处是显而易见的，地图的完整性会直接影响它的可信度。但该改进也可能带来数据冗余以及准确度降低等。

- 规范命名。前面也提到，在地点名称方面，无意义、歧义的名称普遍存在，因此需要根据实际情况加以规范。另外还存在同类地点名称不同等，例如地名“中国工商银行”在地图中的标注就有“ICBC”“工商银行”等，对查询带来不便。该改进对分类查询十分有益，可以让用户更加清楚地了解地图呈现的信息。但在具体操作中，到底哪个命名才能算作“规范命名”，每个人都可能有不同的理解，要保证数据的一致性是有难度的。

- 在用户贡献程度分析中，贡献程度大的用户可能是电脑或者智能设备，也可能是一些忠实用户。我们可以认为，贡献程度大的用户通过程序进行贡献，贡献程度小的可能是通过手动输入进行贡献的。由于手动输入的方式难以规范统一，我们可以考虑删除掉贡献度小的用户。这个举措的好处是可以提高现有数据的准确性与一致性，缺点是删除了一部分的有用信息，并且打击了零散用户的积极性。

额外的数据探索

在数据集中数量最多的十家银行：

```
c.execute('''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;''')

o = c.fetchall()
for row in o:
    print row[0], row[1]
```

```
中国工商银行 16
中国银行 14
中国农业银行 10
招商银行 10
汉口银行 6
中国建设银行 4
工商银行 3
中国民生银行 2
广发银行 2
建设银行 2
```

由图也可以看出命名不规范的负面影响，例如“工商银行”和“中国工商银行”被列为了不同的分组。事实上，如果没有前面对繁简体和中英文的数据清理，同一银行被分到不同类的现象会更多。但该查询同样为我们提供了不少信息。总体而言，工商银行的数量依然是最多的。

结论

此次数据整理工作是比较有成效的。虽然只修改了整个数据集的一小部分，但依然是一个有价值的贡献。我们可以看到，武汉市的数据集存在大量缺失，数据的不规范现象十分严重。因此要完整真实地呈现地理状况，还需要更加细致的努力。