整理 OpenStreetMap 数据

一、地图区域：中国北京市

- https://mapzen.com/data/metro-extracts/metro/beijing_china/
- https://www.openstreetmap.org/relation/912940#map=8/40.255/116.463

研究生就读于北京，对于北京有一种特殊的感情，所以更有兴趣探索北京的地图数据集。

二、地图中存在的问题

我在下载了北京市样本地图以后，对地址进行审查时，主要发现了以下问题：

- 某些地址的英文名称中，存在过度简写的情况。如：“Aimin Str”。
- 某些地址的英文名称中，存在拼音直译的情况。如：“Anding Lu”。
- 某些地址的英文名称中，存在首字母大小写不统一的情况。如：“Street”和“street”

三、数据清理

在对地址的英文名称进行审查的过程中发现，有很多名称存在过度简写、简写不统一及首字母大小写不统一等情况，例如同样是表达“Street”，数据集中共使用了“Str”、“St”、“street”等多种表达。为纠正这一问题，定义了如下修正函数：

```
# 根据审查数据确定mapping
mapping = {
    'Str': 'Street',
    'St': 'Street',
    'street': 'Street',
    'Rd': 'Road',
    'road': 'Road',
    'Lu': 'Road',
    'lu': 'Road',
    'Bld': 'Buiding',
    'Bldg': 'Buiding'
}

def update_name(name, mapping):
    """将地址名以mapping键值结尾修改为mapping对应的value"""
    name_to_be_fixed = mapping.keys()
    for word in name_to_be_fixed:
        if name.endswith(word):
            name = name.replace(word, mapping[word])
    return name

def fix_name(street_name, mapping):
    """匹配数据集中结尾可能不规范的地址名称，并进行更新"""
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            update_name(street_name, mapping)
```

这一函数将使得所有以 mapping 中 key 键结尾的地址名称均修正为 mapping 中对应的 value 值。

四、数据概述

这一部分包括数据库的基本信息，并使用 SQL 进行归集查询。

（一）文件大小

```
import os
def get_size(file):
    size = os.path.getsize(file)/(1024.0 * 1024)
    convert_to_MB = format(size, '0.2f') + "MB"
    print file.ljust(30, '.'), convert_to_MB
get_size('beijing_china.osm')
get_size('OpenStreetMap.db')
get_size('nodes.csv')
get_size('nodes_tags.csv')
get_size('ways.csv')
get_size('ways_tags.csv')
get_size('ways_nodes.csv')
```

```
beijing_china.osm.............. 192.61MB
OpenStreetMap.db............... 99.67MB
nodes.csv...................... 72.00MB
nodes_tags.csv................. 3.03MB
ways.csv....................... 7.65MB
ways_tags.csv.................. 8.59MB
ways_nodes.csv................. 24.94MB
```

（二）使用 SQL 对数据库进行查询

1. 定义查询函数

```
import sqlite3
def SQL(query):
    db = sqlite3.connect('OpenStreetMap.db')
    c = db.cursor()
    c.execute(query)
    rows = c.fetchall()
    return rows
    db.close()
```

2. 节点的数量

```
SQL("SELECT COUNT(*) FROM nodes;")
```

```
[(910374,)]
```

3. 路径数量

```
SQL("SELECT COUNT(*) FROM ways;")
```

```
[(134803,)]
```

4. 唯一用户数量

```
SQL("""
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e
""")
```

```
[(1958,)]
```

5. 节点中“restaurant”的数量

```
SQL("""
SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE value = 'restaurant';
""")
```

```
[(u'restaurant', 1456)]
```

6. 店面最多的 10 家饭店

```
from pprint_utf import pprint
pprint(SQL("""
SELECT value, COUNT(*)
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value = 'restaurant') i
ON nodes_tags.id = i.id
WHERE key = 'name'
GROUP BY value
ORDER BY COUNT(*) DESC
LIMIT 10;
"""))
```

```
[(u'呷哺呷哺', 10),
 (u'庆丰包子铺', 7),
 (u'永和大王', 7),
 (u'沙县小吃', 6),
 (u'田老师红烧肉', 6),
 (u'桂林米粉', 5),
 (u'马兰拉面', 5),
 (u"Annie's", 4),
 (u'巫山烤全鱼', 4),
 (u'眉州东坡酒楼', 4)]
```

五、额外改进建议

在探索数据集的过程中发现：

- 数据集中大部分的“脏数据”均来源于格式不规范
- 通过 SQL 发现数据中包含的部分信息明显少于真实情况（最明显的如呷哺呷哺、庆丰包子铺在北京的分店要远远高于查询出的数据）

因此，针对以上问题分别提出两点建议：

- 针对第一点，可以通过后台控制数据录入的有效性，不符合规定格式的不予通过
- 针对第二点，是否可以与国内地图数据做得比较好的企业如腾讯地图、百度地图等进行数据共享

益处：通过以上改进，可以有效降低数据集使用者进行数据清洗的耗时，且详尽的数据也有利于提高数据集的使用率。

问题：但此类改进尤其是第二点可能带来一些很明显的问题，就是有可能多个数据库之间存在大量的重复数据，要进行合并清理所需要的工作量异常之大。

六、结论

通过本次项目，让我对数据清理的流程更为熟悉。在进行数据清理的过程，夯实了 Python 的代码基础，学习并练习了 SQL 基础知识。

鉴于本人能力和精力所限，未能对数据集进行更为深入的探索，待未来代码能力有所提升，将对此项目进行全面处理，从而精进个人的数据清洗能力。