# OpenStreetMap 数据分析

吕杨

2017 年 7 月 5 日星期三

## ● 图区选择

我选择了美国内华达州的拉斯维加斯（Las-Vegas）作为分析对象：
https://mapzen.com/data/metro-extracts/metro/las-vegas_nevada/

该区域覆盖了整个拉斯维加斯居民区，OSM-XML 未压缩文件大小为 215MB。拉斯维加斯作为全世界最有名的赌城，从建市到国际化城市仅用了十余年。过去我曾去过澳门，在威尼斯人体验过一掷千金的快乐。未来我将前往拉斯维加斯作为我的人生目标之一。

## ● 项目准备及项目流程

从 OSM 官网处，下载 OSM-XML 原始文件并解压，并用以下 Python 脚本观察原数据：

1. tag_count.py 迭代解析统计 xml 文档中关键节点数据
2. tags.py 标签类型，探索数据中是否存在具有问题的字符
3. user.py 查找这一特定地区有多少唯一用户向地图做出贡献
4. audit.py 审查善街道名数据

使用 osm_to_csv.py 完善街道名数据，解析 OSM XML 文件中的元素，将这些元素从文档形式转换为表格形式，从而能够写入 .csv 文件。然后，这些 csv 文件就可以作为表格轻松地导入 SQL 数据库中。

## ● 探索地图数据遇到的问题

1. “k”标签中包含 “:”，不能将此标签简单忽略，例如
   <tag k="addr:street:name" v="Lincoln"/>
2. 部分街道名称存在缩写，例如
   “Las Vegas Blvd”
3. 邮政编码存在多种表现形式，例如
   “Nevada 89113”，“NV 89191”，“89147-8491”

完善数据如下：

1. 标签“k”值包含“:”，则“:”前面的字符应该设为标记类型，“:” 后面的字符应该设为标记键，若“k”值中包含其他“:”，则应该忽略这些“:”并保留为标记键的一部分。

```
if LOWER_COLON.match(tag['key']):
        tag['type'], _, tag['key'] = tag['key'].partition(':')
```

<tag k="addr:street:name" v="Lincoln"/>
将变成
{'id': 12345, 'key': 'street:name', 'value': 'Lincoln', 'type': 'addr'}

2. 将缩写街道名填充完整

```
expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane",
"Road", "Trail", "Parkway", "Commons", "Apache"]

mapping = { "St": "Street","St.": "Street","Ave": "Avenue","AVE": "Avenue","ave":
"Avenue","apache":"Apache","Ave.":"Avenue","Blvd":"Boulevard","Blvd.":"Boulevard","blv
d":"Boulevard","Rd5" :"Road","Rd":"Road","Rd.":"Road","Cir":"Circle","Dr":"Drive","Ln":"La
ne","Ln.":"Lane","ln":"Lane","parkway":"Parkway","Pkwy":"Parkway","S.":"South","S":"South
"}

def update_name(name, mapping):
    m = street_type_re.search(name)
    if m not in expected:
        if m in mapping:
            name=name.replace(m,mapping[m])
    return name
```

调整后变化：Las Vegas Blvd => Las Vegas Boulevard

3. 修改邮政编码为标准 5 位数

```
def update_zip(elem):

    if elem.tag == "node" or elem.tag == "way":
        for tag in elem.iter("tag"):
            if is_zip_code(tag):
                if len(tag.attrib['v']) != 5:
                    if tag.attrib['v'][0:2] == '96':
                        tag.attrib['v'] = tag.attrib['v'][0:5]
                    else:
                        tag.attrib['v'] = tag.attrib['v'][-5:]
```

调整后统一为“89”开头 5 位整数。

- 数据概述

整体数据集大小如下：

las-vegas.osm ..... 215 MB
lasvegasOSM.db ..... 121 MB
nodes.csv ..... 84.3 MB
nodes_tags.csv ..... 2.29 MB
ways.csv ..... 6.47 MB
ways_tags.csv ..... 29.8 MB
ways_nodes.cv ..... 14.3 MB

1. 节点的数量

   sqlite> SELECT COUNT(*) FROM nodes;

   1049632

2. 途径的数量

   sqlite> SELECT COUNT(*) FROM ways;

   112182

3. 唯一用户的数量

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

   1044 个唯一用户。

4. 前 10 的贡献用户

   ```
   sqlite> SELECT a.user, COUNT(*) as num
           FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) as a
           GROUP BY a.user ORDER BY num DESC LIMIT 10;
   ```

   alimamo|251486
   tomthepom|121172
   woodpeck_fixbot|70800
   alecdhuse|66523
   abellao|55608
   gMitchellD|44621
   robgeb|41005
   nmixter|40093
   TheDutchMan13|39262
   Tom_Holland|33028

- 补充想法

  1. 公共设施数

     ```
     sqlite> SELECT value, COUNT(*) as count
               FROM nodes_tags
             WHERE key = 'amenity'
             GROUP BY value
     ```

```
        ORDER BY count DESC
        LIMIT 10;
```

```
restaurant|468
place_of_worship|294
fuel|282
fast_food|267
fountain|266
school|207
shelter|122
toilets|86
café|78
bar|75
```

2. 拉斯维加斯酒店数

```
sqlite> SELECT count(id) FROM nodes_tags where value = 'hotel';
```

57 家酒店。

3. 餐厅风格及数量

```
sqlite> SELECT value,count(*)as num FROM nodes_tags
        WHERE key = 'cuisine'
        GROUP BY value
        ORDER BY num DESC limit 10;
```

```
burger|91
mexican|54
pizza|37
american|25
coffee_shop|20
sandwich|20
chinese|19
italian|17
steak_house|16
chicken|10
```

4. 分析用户对于 OSM 的贡献

```
sqlite> SELECT user,cast(num as float)/cast((select sum(num) from count_per) as float)
as per
          FROM count_per
          ORDER BY per DESC limit 10;
```

alimamo|0.216459777554755
tomthepom|0.104295524068396
woodpeck_fixbot|0.060939186479075
alecdhuse|0.0572578743241173
abellao|0.0478630830752599
gMitchellD|0.038406319772356
robgeb|0.0352939455024642
nmixter|0.0345089661512084
TheDutchMan13|0.0337937053607548
Tom_Holland|0.0284279583478939

在拉斯维加斯区域内，贡献量排名第一的用户占整个用户数据的 21.6%；贡献量前 5 的用户贡献了 48.7%；贡献量前 10 的用户贡献了 65.7%。这显示少量的数据贡献者对地图的整体数据影响较大，而 xml 文档内有大量的贡献数据较少的用户，对于这部分用户提供的数据需要加强对其上传数据的审核，或者设定标准对于达不到一定贡献量的用户数据不予采用。

*益处及可能出现的问题

较少的用户将提供更为规范标准及统一的地图信息，从而将有可能很大程度上减少了地图数据错误及数据清理所需的时间。

但开源地图的数据往往来自活跃用户的自发数据上传，对于数据量少的用户不予采用将可能极大的挫败其积极性及活跃度，对于以用户为核心的社区组织将造成用户的大量损失，从而造成部分析取数据的不完整。

5. 最大宗教集会区域

```
sqlite>SELECT nodes_tags.value, COUNT(*) as num
        FROM nodes_tags
        WHERE nodes_tags.key='religion'
        GROUP BY nodes_tags.value
        ORDER BY num DESC
        LIMIT 1;
```

christian|275

没有意外基督教堂是最大宗教集会区域。

## ● 总结

本项目通过 Python 和 SQLite 工具处理拉斯维加斯市的 OSM-XML 数据，其中完善了节点、街道名和邮政编码数据，并整理了复杂的节点和途径数据，从而顺利解析 XML 文件中的元素，将这些元素从文档形式转换为表格形式，写入 .csv 文件中，方便数据库的加载和分析。

在数据清理过程中发现，开源地图仍旧存在着数据不规范统一的地方，由于来源于众多的数据提供者，街道名和邮政编码出现了明显的不同数据格式或标准，所以导致了前期大量时间用于该内容的数据清理。

取决于未来地图的使用方，在这份 xml 文件中静态公共设施数据源较多，但公共交通内

容较少还需要进一步的完善和补充。

- 附录补充

本项目中，创建数据库、导入 csv 文件以及创建表结构代码如下

```
$sqlite3 lasvegasOSM.db
sqlite> .separator ‘,’
sqlite> .import nodes.csv nodes
sqlite> .import nodes_tags.csv nodes_tags
sqlite> .import ways.csv ways
sqlite> .import ways_tags.csv ways_tags
sqlite> .import ways_nodes.csv ways_nodes

CREATE TABLE nodes (
    id INTEGER PRIMARY KEY NOT NULL,
    lat REAL,
    lon REAL,
    user TEXT,
    uid INTEGER,
    version INTEGER,
    changeset INTEGER,
    timestamp TEXT
);

CREATE TABLE nodes_tags (
    id INTEGER,
    key TEXT,
    value TEXT,
    type TEXT,
    FOREIGN KEY (id) REFERENCES nodes(id)
);

CREATE TABLE ways (
    id INTEGER PRIMARY KEY NOT NULL,
    user TEXT,
    uid INTEGER,
    version TEXT,
    changeset INTEGER,
    timestamp TEXT
);

CREATE TABLE ways_tags (
```

```
    id INTEGER NOT NULL,
    key TEXT NOT NULL,
    value TEXT NOT NULL,
    type TEXT,
    FOREIGN KEY (id) REFERENCES ways(id)
);

CREATE TABLE ways_nodes (
    id INTEGER NOT NULL,
    node_id INTEGER NOT NULL,
    position INTEGER NOT NULL,
    FOREIGN KEY (id) REFERENCES ways(id),
    FOREIGN KEY (node_id) REFERENCES nodes(id)
);
```