OpenStreetMap Data 案例分析

Map Area:

Seattle,WA, United States

https://mapzen.com/data/metro-extracts/metro/seattle_washington/

数据整理过程中遇到的挑战？

数据整理过程中遇到的最大的挑战是不会应用编程技能来进行相关的数据整理工作，不熟悉如何通过使用正则表达式来匹配相关的字符内容从而筛选出符合要求的数据信息。另外一点挑战是关于数据集中的 Dirty Data 的清理工作，由于 OpenStreetMap 网站提供的数据是由不同的用户自发修改整理的非官方的数据，所以数据的格式有时候并不统一，需要自行进行清理。

Problems Encountered in the Map

- Node tags 中的 access 的 value 值不统一，出现了 yes, no, permissive, 甚至 forestry 等多个值，造成混乱
- Node_tags 中的 Key 值有很多自定义的含义不明确的值，例如 node id 29546940 存在着 af, am, ar, bg, bs 等多个含义不明显的 key 标签，会对数据的分析造成困难
- 同样含义的 key 标签会有不同的标志 例如 areaway/aerialway electricity/electrical_appliances
- 同样的 value 值可能对应多个 key 值 例如 WA 表示华盛顿州，可能对应 state state_code gnis:ST_num 多个不同的 key 值
- Postcode /Postal_code 存在多个 KEY 值，而且 Value 不规范，例如 V8W, 2L4, V8T4Y3。

Solutions：

```
mapping = { "electrical_appliances": "electricity",
            "aerialway": "aeroway",
            "state": "state_node",}

def updata_name(name, mapping):
    changeword = mapping.keys()
    for word in changeword:
        if word in name:
            name = name.replace(word, mapping.get(word))
            break
    return name

```

在数据导出之前先对数据进行一定的清洗，将 key 值为 state, aerialway, electrical_appliances,的标签值分别替换为 electricity, aeroway, state_node。

Data Overview

File sizes

Seattle.osm	1642MB
project.db	994MB
nodes.csv	616MB
nodes_tags.csv	42MB
ways_nodes.csv	126MB
ways_tags.csv	49MB

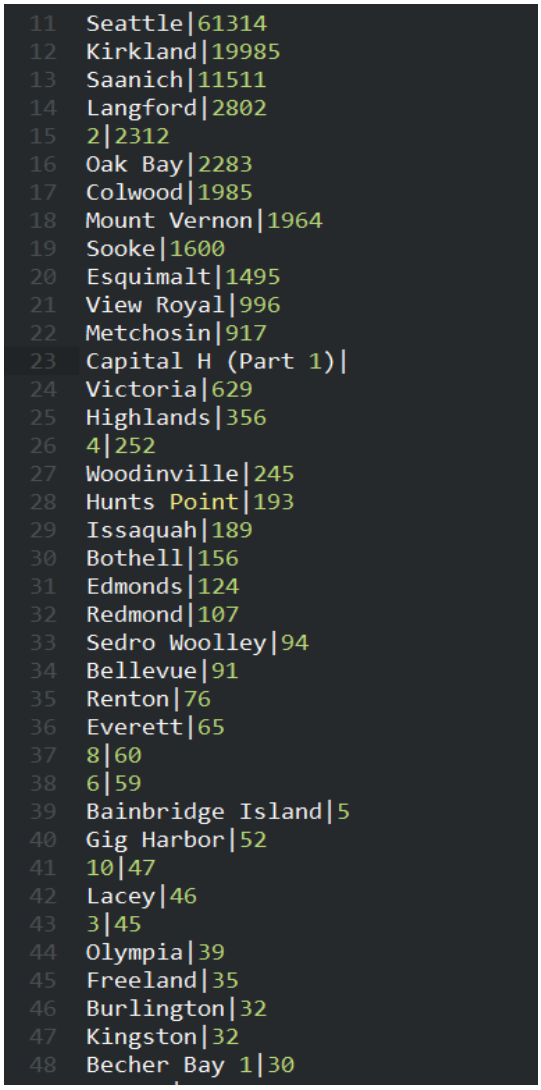
```
#结点的数量
SELECT COUNT(*) FROM nodes;
7318958

#道路的数量
SELECT COUNT(*) as count FROM ways;
452380
```

```
#数据contributor 的数量
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
3091
```

查询各个城市出现的次数

```
SELECT value, COUNT(*) as count
FROM nodes_tags
WHERE key LIKE '%city'
GROUP BY value
ORDER BY count DESC;
```

得到的部分结果，删除了出现次数过小的数据



查询各种不同类型的 shop 出现次数

```
#各种商店的种类名称
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
WHERE nodes_tags.key ='shop'
GROUP BY nodes_tags.value
ORDER BY num DESC;

convenience|705
hairdresser|526
clothes|459
beauty|425
car_repair|304
supermarket|273
yes|208
furniture|180
dry_cleaning|155
mobile_phone|148
car|135
pet|131
bakery|120
gift|116
books|108
optician|108
art|99
shoes|92
electronics|87
tobacco|87
jewelry|85
sports|84
car_parts|82
bicycle|79
wine|79
alcohol|78
department_store|66
doityourself|64
hardware|56
massage|56
laundry|53
garden_centre|50
copyshop|47
```

Suggestions about improving data quality

作为开源的项目，应当在每一个新加入的 contributor 做出改动之前给他们提供一个具体的指导文档，来规范他们对数据集的修改，例如可以考虑对于每一个 node 结点都设定固定的 Key 值模板，比如给出 highway, address, country, layer 等固定的 key 值，同时对于不在模板中的 key 值，可以单独保留，定期由人工进行审核复查来判断是否可以导入到新的数据文件中，以此来尽量减少不一致并且有重合的 key 值。预期会出现的问题是模板给出的 Key 值并不能够全面覆盖所有结点的相关属性结点属性，而且随时间变化，会有一些新出现的特殊属性值标签，比如新兴的共享单车专用停车区可能就是一个不同于传统的自行车停车区的新属性值，需要对其进行单独的标注，所以可以考虑正则表达式规则来规范每个 key 值的基本格式并结合人工审查的机制。同时可以考虑给予忠实的用户更大的操作权限，而限制新用户的数据编辑权限，这样虽然可能会减少新用户的编辑问题，但是可以使得数据格式更加规范，而且根据统计结果显示，二八原则是使用的，接近 2 成的忠实用户贡献了 8 成左右的数据。

结合使用 excel 和 SQLite，使用 excel 可以在将数据导入到 SQLite 之前对于数据的基本概况有一定了解，比

如缺失值，数据的类型，数据的特征，从而辅助后面使用 SQLite 进行精确的查找。而数据库也是处理大量数据时不可或缺的工具，因为数据库的确比 excel 在查找数据上又更加强大的功能，首先 csv 文件在 excel 中最多显示 1048575 行，数据量过大时,excel 不能显示全部的数据，其次，在 SQLite 中，不同的数据表格之间可以很方便的 join，以此来分析有相关性的数据，最后 SQLite 的强大的查询可以根据要求进行更加精确的查找。

关于 relation 区域的数据不太充足，可以考虑将城区进行更加细致合理的区分和规划，突出每一个区域的功能特点。比如学校，商业区，工业区可以进行特定的表示，同时都可以整合 nodes 和 ways 标签的具体的属性值的特点来进入深入的分析，或许可以指导城市的规划问题，探究解决城市拥堵，安排布局不合理的问题。但是因为功能区的整体特征并没有一个概论，会有功能区的定位模糊的问题存在，可能某个区域既有工业区也有商业区等问题，处理难度较大。