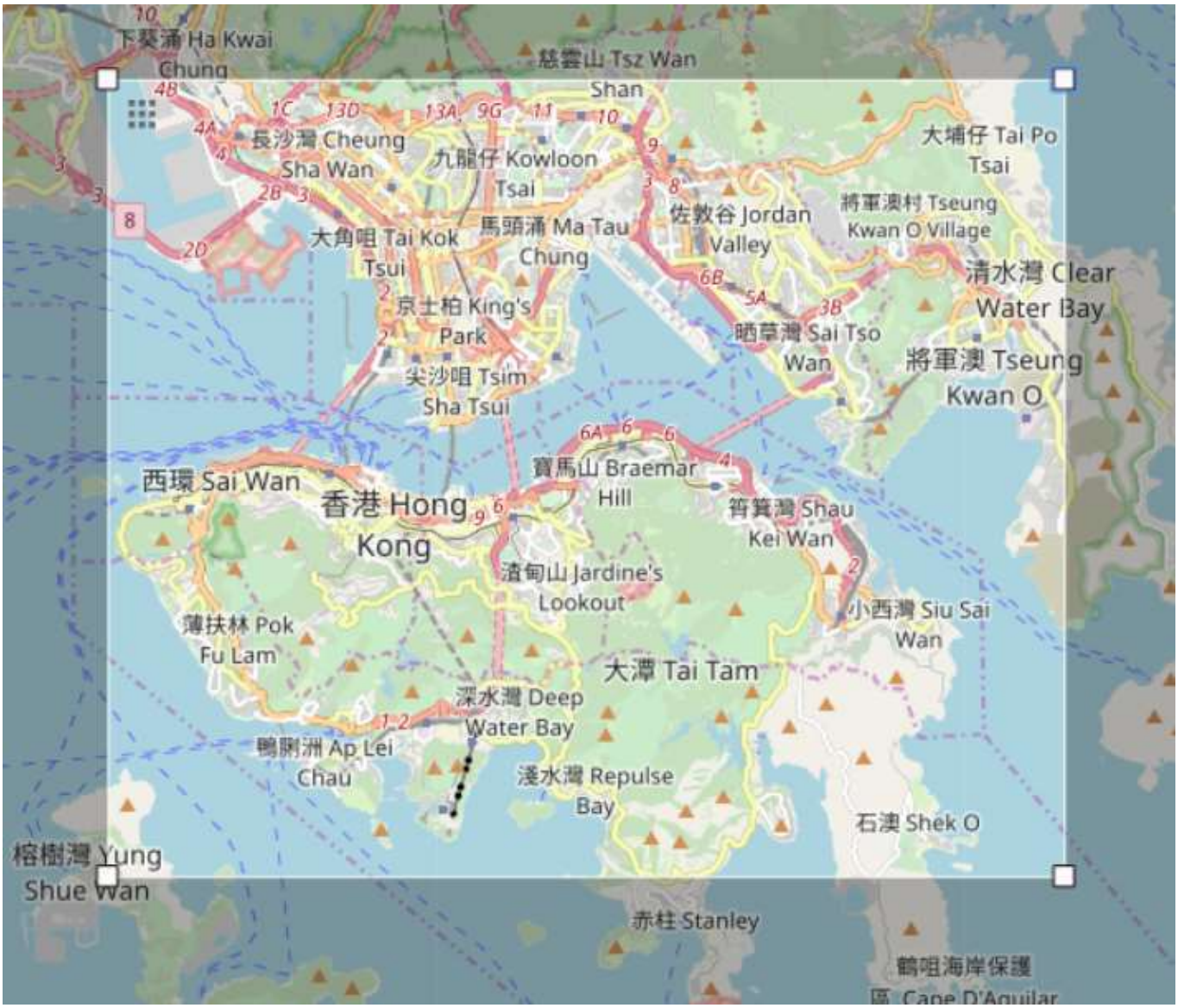
OpenStreetMap 数据整理实践

1. 选定区域

Hong Kong, China

- https://www.openstreetmap.org/relation/913110

选取的区域是香港的一部分， `hong-kong.osm` 大小 92.6M。经纬度边界：22.2233, 114.115, 22.3475, 114.275，以下为选定区域示意图：



2. 选择区域的原因

香港是一个神奇的城市，你可以在这里享受购物的快乐、世界各地的美食，也可以体验独特的香港文化。很喜欢在香港的街头闲逛，总是会有很多惊喜。那么，如果在香港街道数据中“闲逛”，又会发现什么呢？

3. 地图数据中的问题

我将下载后的 `hong-kong.osm` 通过 `make_a_simple.py` 代码，转换为一个小型的 `sample.osm` 文件，对数据进行审查。

3.1. key 审查

- 使用 `tags.py` 审查没有发现问题 key（problemchars 类型），将 other 类型的 key 打印出来发现有一部分是因为有 2 个冒号，除去两个冒号的，其他有：
  - **`naptan:Bearing`**：在 NaPTAN Wiki 中搜索到相关信息，**`Direction along street in which vehicle is pointing when stopped at stopping point.`** 表示的是：车辆停在这个位置时，车头的朝向方位是什么。
  - **`ref:CBNW`**：在 Key:ref Wiki 上可以查到，`ref:CBNW` 更通用的写法应该是 `ref:ctb`，可以修正一下，不过对本次查询没有影响，暂时未修改；
  - **`socket:bs1363`**：在 charging_station Wiki 是充电站正常的 key，表示的是插座的一种类型；
  - **`name:zh-yue`**：应该代表的是粤语，但是没有找到对应的 Wiki，而且中文或者粤语，文字应该是没有区别的，香港都是使用的繁体字，zh-yue 似乎没有必要，只是在下方处理的时候需要注意；
  - **`boundary_1`**：关于边界，没有找到 boundary_1 相关信息，暂时忽略；
  - **`NRG`**：没有找到相关信息，也暂时忽略。
- 整理数据后想要探索的相关 key：
  - 街道地址：**`addr:street`**
  - 生活设施：**`amenity`**
  - 餐厅：**`restaurant`**
  - 菜系：**`cuisine`**，这其中发现一个问题是，cuisine 的值如果有多个，被用分号分开了，所以需要处理一下。

3.2. 街道地址的审查

- 因为历史原因与地域原因，香港是一个国际化的大都市，常用语言有好几种：中文、粤语、英文、葡文等。所以地址的名称一般情况下都是中英文结合并且是先中后英的形式；
- 很多 `name` 的 key 后面都跟着几个 `name:zh`，`name:en` 等区分语言的名称（这里多处理一步，如果没有 `name:zh`，也要搜索一下 `name:zh-yue`）；
- 后续可以整理查看 `name:zh`，`name:en` 的值是否都在 `name` 中，如果没有，则将 `name` 修改为 `name:zh` 与 `name:en` 用空格连接的字符串
- 通过使用 `audit.py` 进行审查发现地址的格式还是比较整齐的，只修改了一个以 Roadww 结尾的地址为 Road 结尾。

4. 问题数据的改进

4.1. cuisine 菜系数据的 value 处理

处理方式：

- 在写入 csv 文件的时候，查找 cuisine 值中包含分号的值；

- 用分号将这个值分割为一个列表；
- 对列表中的项进行遍历，每一项创建一个新的 tag，key 为 cuisine，value 为正在遍历的项

以下代码节选自 `data.py` 主要用来处理多个 `cuisine` 的情况

```
def multi_cuisine_apart(tag_dict):
    """
    将有多个菜系的 cuisine，用 ； 分开，整理为一个列表
    如果只有一个菜系，则为一个元素的列表
    最终返回这个菜系列表 cuisine_list
    """
    cuisine_list=[]
    if ";" in tag_dict['value']:
        cuisine_list = tag_dict['value'].split(';')
        # print cuisine_list
    else:
        cuisine_list.append(tag_dict['value'])
    return cuisine_list
```

..... 在 `data.py` 的 `shape_element` 函数中，将原来的 `tags.append(tag_dict)` 改为了以下语句

```
if (tag_dict['key'] == 'name') and zh_en_name:    # 如果 key 为 name，并且 zh 和 en 都不为空
                    tag_dict["value"] = zh_en_name                # 则修改 name 对应的 value

                if tag_dict['key'] == 'cuisine':                   # 处理 key 为 cuisine 的数据
                    cuisine_list = multi_cuisine_apart(tag_dict)
                    for cuisine in cuisine_list:
                        tag_dict_cuisine = tag_dict.copy()
                        tag_dict_cuisine["value"] = cuisine
                        tags.append(tag_dict_cuisine)
                else:
                    tags.append(tag_dict)
```

4.2. name 与 name:zh、name:en 的结合

处理方式：

- 遍历 node
- 判断该 nodeid 中是否有 name:zh 和 name:en
- 如果满足以上条件则给 name key 对应的 value 赋值为：name:zh 的值 + 一个空格 + name:en 的值

以下代码为 `data.py` 的 `find_zh_en_name` 函数

```
def find_zh_en_name(elem):
    """
    name:zh 和 name:en 将被分别储存在 zh 和 en 中并最终组合为 zh_en_name 返回
    同时返回值还有 has_name 用来解决有 name:zh 和 name:en 却没有 name key 时的情况
    """
    zh = ""
    en = ""
    zh_en_name=""
    has_name = False
    for child in elem:
```

```
        if child.tag == "tag":

            if (child.attrib['k'] == "name:zh") or (child.attrib['k'] == "name:zh-yue"):
                # 中文要处理 zh-yue 的情况
                zh = child.attrib['v']
            if child.attrib['k'] == "name:en":
                en = child.attrib['v']
            if child.attrib['k'] == "name":
                has_name = True
    if zh and en:
        zh_en_name = zh + " " + en
    return zh_en_name, has_name
```

……在实现了上述步骤之后，debug 时发现有一些既有 en 又有 zh 名称的 tag，可能没有 name 这个 key，这种情况下，添加了以下代码来处理这个问题，位置在 `data.py` 下面的 `shape_element` 函数中，`for child in element:` 之前：

```
zh_en_name, has_name = find_zh_en_name(element) # 调用函数查看是否有 name, zh 和 en
    if (zh_en_name) and (not has_name):
        # 如果没有 name，但是有 zh 和 en，则创建一个新的 name 条目
        tag_dict_name={}
        tag_dict_name["id"] = node_id
        tag_dict_name["key"] = "name"
        tag_dict_name["value"] = zh_en_name
        tag_dict_name["type"] = default_tag_type
        tags.append(tag_dict_name)
```

4.3. 数据整理中遇到的问题

- 因为对 xml 文件的写入不熟悉，所以直接在写入 CSV 文件的时候进行了整理，所以有一个问题是，如何在 xml 文件中做出这些整理的操作，并写入到 xml 文件中
- 关于 name 的整理中，在实现了上述功能之后，浏览文件时发现：有一些既有 en 又有 zh 名称的 tag，可能没有 name 这个 key，这种情况下，我的解决思路是：需要添加一个 "name" 的 key，并赋予它 zh + " " + en 这个值，并按照正常 name 一样，type 设置为 regular。这部分内容在以上代码中也有所体现。

5. 用 SQL 查询数据

将整个 `hong-kong.osm` 整理为 csv 文件，并导入到 sql 中，代码文件为：

- `data.py` —— 将 xml 写入 csv 文件
- `trans_db.ipynb` —— 将 csv 文件写入数据库

5.1. 文件大小

- hong-kong.osm: 92.6 MB
- mydb.db: 49.8 MB
- nodes_csv: 31.6 MB

- nodes_tags.csv: 3.2 MB
- ways_csv: 2.86 MB
- ways_nodes.csv: 11.4 MB
- ways_tags.csv: 7.02 MB

5.2. node 数量

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

nodes 数量查询结果
400770

5.3. way 数量

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

ways 数量查询结果
50217

5.4.唯一用户数量

```
SELECT COUNT(DISTINCT(sq.uid)) as 'Number of unique users'
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) as sq;
```

唯一用户数量查询结果
907

5.5. 贡献前 10 的用户

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

贡献前 10 用户查询结果
hlaw
KX675
FlyTy
Philip C
R17466
Wrightbus
jc86035
cartogram
eversone
bTonyB

5.6. 排名前 10 的咖啡店

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='cafe') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='name'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 10;
```

前 10 咖啡馆查询结果
星巴克咖啡 Starbucks Coffee
太平洋咖啡 Pacific Coffee
Starbucks
Pacific Coffee
Starbucks Coffee
Délifrance
Pacific Coffee Company
Paradise
新剑記
18 grams

5.7. 规模前 10 的快餐店

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='fast_food') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='name'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 10;
```

快餐前 10 查询结果
麥當勞 McDonald's
McDonald's
大家樂 Café de Coral
吉野家 Yoshinoya
肯德基 KFC
大快活 Fairwood
KFC
Maxim MX
賽百味 Subway
Subway

5.8. 最受欢迎的菜系前 10 名

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC
Limit 10;
```

前 10 菜系查询结果
chinese
japanese
noodle
pizza
indian
noodles
thai
american
sushi
vegetarian

6. 额外的想法

6.1. 更多问题与改进方法

在进行了 SQL 查询之后，发现了更多问题。比如：

1. 有些名称有不同的版本，比如 Starbucks Coffee、Starbucks、Starbucks Reserve 都是星巴克，可以只使用最有标识性的名称，比如 Starbucks，在数据整理时，如果有相关的需求，也可以回过头去获取 name 中包含 Starbucks 的值，全部修改为一致的名称；
2. name 有的还是存在只有英文的情况，比如 麦當勞 McDonald's 和 McDonald's、星巴克咖啡 Starbucks Coffee 和 Starbucks Coffee，已经做了一定的改进，比如查找到有 name:zh 和 name_en 的 node，则将 name 修改为它们的连接字符串，但是很有可能出现没有细分语言的 name 的情况，；
3. 甚至一些有规定样式的，也有不同的用法，比如 noodles 和 noodle，在 Key:cuisine-wiki 上可以看到应该统一为 noodle；
4. cuisine，这个表示的是：For describing the type of food served at an eating place.中文翻译为：用来记录一个餐饮场所的食物风味。但是我们看到 cuisine 的值，既有按地域分类的，比如 chinese、japanese，又有按照食物种类分类的 noodle、pizza，个人感觉这两种并不是同一种分类，比如 noodle 种类，有可能是中餐也有可能是日本拉面，这两种分类是交叉重合的。

6.2. 改进后的预期问题

1. 前 3 条问题都可以归纳为一个问题，那就是贡献者的贡献信息参差不齐。当然可以做出各种规定性的标准，但是类似 noodles 那一项也可以看出，有时候做出了规定，但是贡献值自身如果不小心还是会提供有差异的信息。如果可以像 Github 那样，有更多的志愿者来 review 新贡献者的贡献， review 通过后才可以被添加进去。这样的问题就是可能没有那么多志愿者维护。

2. cuisine，可以添加另一个分类，菜系和食物种类并不相同，可以加一个食物种类的分类，就叫 food，这样可以将菜系和食物分类区分开，但是有可能的问题是，food，包含的东西太多，可能贡献者会提供并不像 noodle 和 pizza 这样有代表性的分类性质的词，而是添加很多无关紧要的食物名称；
3. cuisine，还有一个解决办法是，在 Key:cuisine-wiki 页面上也可以看到： “It was suggested [1] to use culture=* instead of cuisine=* for other ethnic, cultural and/or regional services like hairdresser or clothes.” 这个解决办法比上一条要好，但是还是有一些不够明确的地方，个人觉得，可以只将地域风格作为一个新的 key，这是比较容易区分和定义的，但是在日常生活中，我们确实会将中餐和 pizza 当作并列的选项来提供，如果在 OSM 中，提供了另外的分类，肯定也会引起混乱。

7. 结论

经过本次整理，数据在一致性和完整性上有了一定的提升。虽然整理后的数据依然存在问题，但是也已经对数据整理进行了深入的实践和体会了。

在本次数据整理和思考的过程中，无疑可以看出 OpenStreetMap 的数据确实有很多不规范的地方，这也是开源项目不可避免会出现的问题。如果真的有需要使用这些数据进行数据分析工作，一定要进行很多的数据整理，并且这些工作还可能会随着分析的深入需要反复进行。第一次窥探到现实世界数据分析工作的冰山一角，很有挑战性！