OpenStreetMap Data Wrangling with MongoDB

Joel Huang

Map Area:Taipei, Taiwan, China
OpenStreetMap Link
MapZen OSM Link

Overview

OpenStreetMaps.com 是一個開源的世界地理地圖，它包含的數據從交通，建築，地址，商家，自然環境...。分享檔案格式採用 XML，使人們可以方便地解析此數據，以及它有優異的檔案壓縮比，可使得檔案容易被交流。如此優異的可讀性，也意涵需要時間去解析。

解析檔案前必須確認問題核心，就如楊過久鬥樊一翁，發現樊翳子動向的規律，才把它剪下。解析過程也如此，也必須從最外層的標籤作為起手式，再來到內部標籤。尤其是這是多人分享的數據，在彼此生活環境不同，同物件會有不同的用語，抑或是使用的數據形態不同步，一件事情多種表述，使修復數據就會變成一個耗費時日的工程。

最後使用MongoDB數據庫查詢，來提供台北數據統計概述。

整個文檔內容圍繞在以下

1. 地圖中遇到的問題
2. 數據概述
3. 關於數據及的其他想法
4. 結論

1. 地圖中遇到的問題

家用語差異，就說澳大利亞，street type竟然超過二十種名稱，查詢的名稱常常霧裡看花，boulevard, crescent, alley, trace, terrace, court, run, walk, pike ...不是本地人，很難知道這意思是路的描述，最後還是選擇自己熟悉的城市。
在修復地圖檔案主要的核心問題有以下：

1. 中英同表。
2. 一個中文，各自表述。
3. 誤解意思。

---

1. 中英同表

- 首先叫出興趣項目，查看資料。

```
D = Counter()
for _, element in ET.iterparse(OSMFILE):
    if element.tag == 'node' or element.tag =='way':
        for child in element:
            if child.tag == 'tag':
                if child.attrib['k'] == 'surface':
                    if child.attrib['v'] in D:
                        D[child.attrib['v']] +=1
                    else:
                        D[child.attrib['v']] = 1
D.most_common()
```

out[3]:

```
[('asphalt', 1370),
 ('paved', 549),
 ('unpaved', 165),
 ('concrete', 161),
 ('paving_stones', 136),
 ('sett', 54),
 ('wood', 35),
 ('ground', 34),
 ('earth', 26),
 ('瀝青', 22),
 ('gravel', 12),
 ('compacted', 11),
 ('dirt', 10),
 ('grass', 8),
 ('cobblestone', 8),
 ('grass_paver', 7),
 ('concrete:lanes', 6),
 ('pebblestone', 6),
 ('wooden', 6),
 ('stone_steps', 5),
 ('cement_paving', 3),
 ('asphaltdesignated', 3),
 ('asphalt;paving_stones', 2),
```

列舉出同意思，不同表述的屬性。

- 'asphalt', '瀝青', 'asphaltdesignated'
- 'concrete', 'concrete:lanes','cement_paving'
- 'paved', 'sett', 'stone_steps'

處理方式:建構字典，把相同的映射到同一個名稱。

2.一個中文，各自表述

依照上列代碼，查詢operator的屬性

```
[('一工處', 332),
 ('臺北大眾捷運股份有限公司', 228),
 ('中華郵政', 213),
 ('公/汐止營運所', 136),
 ('公/萬里營運所', 134),
 ('巨大機械工業(股)', 70),
 ('公/東區營業分處', 69),
 ('統一超商股份有限公司', 67),
 ('全家便利商店股份有限公司', 63),
 ('Taiwan Railway Administration', 54),
 ('公/淡水營運所', 33),
 ('台灣高速鐵路股份有限公司', 32),
 ('7-Eleven', 24),
 ('萊爾富國際股份有限公司', 19),
 ('新北市立圖書館', 19),
 ('歐特儀股份有限公司', 18),
 ('中華郵政股份有限公司', 17),
 ('臺灣鐵路管理局', 16),
 ('台灣中油', 16),
 ('臺北市政府 Taipei City Government', 15),
 ('臺北市公車聯營管理中心', 11),
 ('新北市政府教育局', 11),
 ('台亞石油股份有限公司', 10),
 ('萊爾富國際', 9),
 ('7-11', 9),
```

很多同樣商家，機構，卻有者不同的名稱

- '統一超商股份有限公司', '7-Eleven', '7-11'
- '中華郵政', '中華郵政股份有限公司'

處理方式：選擇統計多數的去做字典映射，少數統計上只有一次的給予忽略。

3. 誤解意思

查詢，台北前十大的店舖

```
pipeline = [{ "$match" : { "Tour.shop" : { "$exists" : 1 },
                           'Tour.shop':{'$ne':None}} },
            { "$group" : { "_id" : "$Tour.shop",
                           "count" : { "$sum" : 1 } } },
            { "$sort" : { "count" : -1} },
            { "$limit" : 10 } ]
```

out[4]:

```
{u'_id': u'convenience', u'count': 2312}
.
.
.
.
{u'_id': u'yes', u'count': 212}
.
.
.
```

台北沒有店名叫做 **yes**，解讀是，數據分享者，在填寫資訊時候，把shop 屬性誤解成**shop存在與否**。

2. 數據概述

這裡介紹檔案的大小，數據庫使用MongoDB Query 去查詢數據的統計。

檔案大小(使用Terminal)

ls -lh taipei_taiwan.osm
174M

ls -lh taipei_taiwan.osm.json
222M

有多少筆文檔(pymongo)

`

collection.find().count()
`

900887

Node and Way 有多少筆

```
pipeline = [{"$ group":{"_id": "$ type",
                        "count": { "$sum" : 1 }  } } ]
```

{'_id': 'way', 'count': 103162}
{'_id': 'node', 'count': 797725}

台北有幾家7-Eleven

```
pipeline =
[{"$ match":{"Daily.operator":"7-Eleven"}},
 {"$ group":{"_id":"7-Eleven",
             "count":{ "$sum":1}}}]
```

{u'_id': u'7-Eleven', u'count': 37}

台北前十大的店舖

```
pipeline = [ { "$match" : { "Tour.shop" : { "$exists" : 1 },
                            'Tour.shop':{'$ne':None}} },
             { "$group" : { "_id" : "$Tour.shop",
                            "count" : { "$sum" : 1 } } },
             { "$sort" : { "count" : -1} },
             { "$limit" : 10 } ]
```

{u'_id': u'convenience', u'count': 2312}
{u'_id': u'clothes', u'count': 677}
{u'_id': u'supermarket', u'count': 460}
{u'_id': u'hairdresser', u'count': 390}
{u'_id': u'bakery', u'count': 355}
{u'_id': u'beverages', u'count': 328}
{u'_id': u'yes', u'count': 212}
{u'_id': u'chemist', u'count': 189}
{u'_id': u'motorcycle', u'count': 185}
{u'_id': u'books', u'count': 180}

前十大餐廳

```
pipeline = [ { "$match" : { "Tour.cuisine":{'$ne':None}}},
                { "$group" : { "_id" : "$Tour.cuisine",
                   "count" : { "$sum" : 1 } } },
             { "$sort" : { "count" : -1 } },
             { "$limit" : 10 } ]
```

{'_id': 'chinese', 'count': 391}
{'_id': 'japanese', 'count': 179}
{'_id': 'coffee_shop', 'count': 177}
{'_id': 'burger', 'count': 135}
{'_id': 'regional', 'count': 79}
{'_id': 'breakfast', 'count': 73}

{'_id': 'italian', 'count': 69}
{'_id': 'pizza', 'count': 40}
{'_id': 'sandwich', 'count': 37}
{'_id': 'asian', 'count': 27}

台北常見的營業時間

```
pipeline = [ {'$match':{'Tour.opening_hours':{'$ne':None}}},
             { "$group" : { "_id" : "$Tour.opening_hours",
                            "count": {"$sum": 1 }}},
             { "$sort" : { "count" : -1 }},
             { "$limit" : 5 }]
```

{u'_id': u'24/7', u'count': 631}
{u'_id': u'Su-Mo 09:00-17:00; Tu-Sa 08:30-21:00', u'count': 50}
{u'_id': u'Mo-Su 11:00-22:00', u'count': 26}
{u'_id': u'Mo-Fr 09:00-15:30', u'count': 23}
{u'_id': u'Mo-Su 10:00-22:00', u'count': 22}

書店常見營業時間

```
pipeline = [ {'$match':{'Tour.shop':'books'}},
             { "$group" : { "_id" : "$Tour.opening_hours",
                            "count" : { "$sum" : 1 } } },
               { "$sort" : { "count" : -1 } },
               { "$skip" : 1 },
               { "$limit" : 1 } ]
```

{u'_id': u'Mo-Su 10:00-22:00', u'count': 3}

台北前十大宗教

```
pipeline = [{"$match":{"Daily.religion":{"$exists":1}}},
            {"$group":{"_id":"$Daily.religion",
                       "count":{"$sum":1}}},
            {"$sort":{"count": -1}},
            {"$limit":5}]
```

{u'_id': u'taoist', u'count': 303}
{u'_id': u'christian', u'count': 253}
{u'_id': u'buddhist', u'count': 169}
{u'_id': u'hindu', u'count': 5}
{u'_id': u'muslim', u'count': 4}

- taoist:道教

時速超過60 Km/Hr的道路是什麼

```
pipeline = [{'$match':{'Daily.maxspeed':{'$gt':60},
                       'Daily.surface':{'$ne':None} }},
            {"$group":{"_id":"$Daily.surface"}}]
```

{u'_id': u'asphalt'}

- 瀝青柏油路。

3.關於數據及的其他想法

```
{u'_id': u'bank', u'count': 844}
```

這個數據結果，讓我覺得不可思議，本者對數據的起疑，先假設地址是否跨區，ＧＰＳ精細度不高，是否同銀行。

- 首先調閱地址登記城市。

```
{'_id': None, 'count': 684}
{'_id': '臺北市', 'count': 54}
{'_id': '台北市', 'count': 31}
{'_id': '新北市', 'count': 26}
{'_id': '新北市新莊區', 'count': 16}
{'_id': '新北市板橋區', 'count': 7}
{'_id': '新北市新店區', 'count': 7}
{'_id': '桃園市龜山區', 'count': 4}
{'_id': '新北市五股區', 'count': 3}
{'_id': '新北市樹林區', 'count': 2}
```

從調閱出的數據，發現幾件事情。

1. None 高達六百多筆
2. 台北，臺北，數據是否重疊
3. 數據統計到了，外圍的新北市
4. 新北市數據，又細分了「區」，這些數據是否也重疊新北市

- 調閱GPS紀錄

```
{'_id': None, 'count': 31}
{'_id': [25.0079598, 121.4597183], 'count': 2}
{'_id': [25.0081238, 121.4602451], 'count': 2}
{'_id': [25.0434862, 121.5157333], 'count': 1}
{'_id': [25.0522781, 121.5170919], 'count': 1}
{'_id': [25.0325916, 121.5165194], 'count': 1}
{'_id': [25.0567881, 121.5096623], 'count': 1}
```

在GPS上，很少資訊提供者，極少使用衛星定位，大幅增加了數據重複可信度。

- 查詢銀行所在地址

```
{'_id': '中正路', 'count': 22}
{'_id': '松江路', 'count': 13}
{'_id': '重慶南路一段', 'count': 12}
{'_id': '中山路一段', 'count': 10}
```

我們查詢谷歌，中正路上銀行，結果是不超過十家，從圖上倒是有一個數據吸引了我，谷歌把**自動提款機ATM**列在地圖上。

4. 結論

台北OSM數據明顯的不完整，從以上例子，銀行數據，出現了用用字不同導致「重複」，「跨區」，有自動提款機設定成銀行的可能。

原本可以從其他特徵，確認是否為銀行，像是地址登記是最清楚的，可是缺乏地址數據，或是landuse判斷是否在商業區，抑或是從Fire Hydrant 是否屬於牆掛，推判出這是否為大樓。

假使，ＧＰＳ經準度是高的，中文字問題得到改善，可以預期的，把這問題模組化，用周遭的數據當作特徵，用機器學習分類算法Naive Bays Classifier，是一個可研究改善的項目。