OpenStreetMap Project Data Wrangling with MongoDB

Wang Xiaochu

Map Area

Birmingham, Westmidlands, England

- OpenStreetMap_Birmingham
- MapData_Birmingham

This city is where I studied in the university for a master's degree of urban and reginal planning last year. So study the city from a different perspective using new skills I've learned means a lot to me.

Problems Encountered in the Map

After initially downloading a small sample size of the Birmingham area and running it against a provisional data.py file, I noticed three main problems with the data, which I will discuss in the following order:

- Overabbreviated & Misspelled street names ("Roxburgh G4rove", "Hidcote Aveune")
- "Incorrect" postal codes (Birmingham postcodes all begin with "B" however a large portion of all documented postcodes were outside this region.)

Overabbreviated & Misspelled Street Names

Before the data was imported into MongoDB, it was audited in audit.py using the following function:

```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane", "Road",
        "Trail", "Parkway", "Commons", "Way", "Walk"]

def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)

def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")

def audit(osmfile):
    osm_file = open(osmfile, "r")
    street_types = defaultdict(set)
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    osm_file.close()
    return street_types
```

This basic querying revealed street name abbreviations and misspellings. I updated all substrings in problematic address strings, such as:
"Aveune" becomes "Avenue", "G4rove" becomes "Grove".

Postal Codes

Postcodes turn out to be a non-technical problem. It revealed some of the data out of the range of Birmingham city, which force me to filter out all of nodes with postcodes start with "C" or "D" in the process of import data into JSON file with adding following codes:

```
if m == 'addr:postcode' and elem.attrib['v'][0] != 'B':
    return None
```

Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

File Size

birmingham_england.osm …….. 1,535 MB

birmingham_england.osm.json …. 1,780 MB

- **Number of documents**

```
> db.birm.find().count()

7017662
```

- **Number of nodes**

```
> db.birm.find({"type":"node"}).count()

6906846
```

- **Number of ways**

```
> db.birm.find({"type":"way"}).count()

110816
```

- **Number of unique users**

```
> db.birm.distinct("created.user").length

2249
```

- **Top 10 contributing user**

```
> db.birm.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                   {"$sort":{"count":-1}}, {"$limit":10}])

{ "_id" : "brianboru", "count" : 3549542 }
{ "_id" : "blackadder", "count" : 447958 }
{ "_id" : "Miked29", "count" : 370951 }
{ "_id" : "James Derrick", "count" : 311132 }
{ "_id" : "mrpacmanmap", "count" : 164581 }
{ "_id" : "Curran1980", "count" : 136896 }
{ "_id" : "PeterP", "count" : 115926 }
{ "_id" : "srbrook", "count" : 105316 }
{ "_id" : "The Maarssen Mapper", "count" : 104912 }
{ "_id" : "richardwest", "count" : 104404 }
```

- **Number of users appearing only once (having 1 post)**

```
> db.birm.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                   {"$group":{"_id":"$count", "num_users":{"$sum":1}}},
                   {"$sort":{"_id":1}}, {"$limit":1}])

{ "_id" : 1, "num_users" : 365 }
```

Additional Ideas

Region statistics and suggestion

Although some of nodes out of Birmingham region have been filtered by their postcodes, there are still indiscriminable nodes do not have postcode attribute left in the file.
Using the follow ing query, I explore the statistics further:

```
> db.birm.aggregate([{"$match":{"addr.city":{"$exists":1}}},
                   {"$group":{"_id":"$addr.city","count":{"$sum":1}}},
                   {"$sort":{"count":-1}}])

{ "_id" : "Birmingham", "count" : 3884 }
{ "_id" : "Solihull", "count" : 910 }
{ "_id" : "Bromsgrove", "count" : 238 }
```

```
{ "_id" : "Alcester", "count" : 158 }
{ "_id" : "Sutton Coldfield", "count" : 106 }
{ "_id" : "Tipton", "count" : 63 }
{ "_id" : "Madeley", "count" : 49 }
.....
{ "_id" : "Ironbridge", "count" : 26 }
{ "_id" : "Wolverhampton", "count" : 22 }
{ "_id" : "bm", "count" : 16 }
{ "_id" : "West Bromwich", "count" : 16 }
{ "_id" : "Redditch", "count" : 13 }
```

It turns out I didn't wrangle this data properly.
One way to solve this problem could be comparing the longitude and latitude of Birmingham city and the attribute of each node.
The advantage of doing so is it will produce more accurate results, and help making better decisions. This method will work well if it was in the context of rigid city analysis. While in this exercise, it can be costly and time-consuming.

Another way is to ensure the property “addr.city” is properly informed, which is an easy-to-implement method.
However, whether this method will work well depends on the integrity of property of city. And lacking of this property in the dataset indicates inaccurate results, which will work the same as postcodes method previously used.
In the circumstance, rather than filter out outer cities incomplete, treat them as parts of big region of Birmingham metropolitan area should be easier and better.

Additional data exploration using MongoDB queries

- **Top 10 Contributers**

```
> db.birm.aggregate([{"$group":{"_id":"$created.user",
                                "count":{"$sum":1}}},
                    {"$sort":{"count":-1}}, {"$limit":10}])

{ "_id" : "brianboru", "count" : 3549542 }
{ "_id" : "blackadder", "count" : 447958 }
{ "_id" : "Miked29", "count" : 370951 }
{ "_id" : "James Derrick", "count" : 311132 }
{ "_id" : "mrpacmanmap", "count" : 164581 }
{ "_id" : "Curran1980", "count" : 136896 }
{ "_id" : "PeterP", "count" : 115926 }
{ "_id" : "srbrook", "count" : 105316 }
{ "_id" : "The Maarssen Mapper", "count" : 104912 }
{ "_id" : "richardwest", "count" : 104404 }
```

- **Most common building types**

```
> db.birm.aggregate([{'$match': {'building': {'$exists': 1, "$ne":"yes"}}},
                    {'$group': { '_id': '$building','count': {'$sum': 1}}},
                    {'$sort': {'count': -1}}, {'$limit': 10}])

{ "_id" : "residential", "count" : 2147 }
{ "_id" : "industrial", "count" : 563 }
{ "_id" : "retail", "count" : 391 }
{ "_id" : "entrance", "count" : 135 }
{ "_id" : "commercial", "count" : 109 }
{ "_id" : "school", "count" : 107 }
{ "_id" : "garages", "count" : 84 }
{ "_id" : "university", "count" : 60 }
{ "_id" : "church", "count" : 52 }
{ "_id" : "storage_tank", "count" : 48 }
```

- **Top 10 appearing amenities**

```
> db.birm.aggregate([{"$match":{"amenity":{"$exists":1}}},
                    {"$group":{"_id":"$amenity","count":{"$sum":1}}},
                    {"$sort":{"count":-1}}, {"$limit":10}])

{ "_id" : "post_box", "count" : 3533 }
{ "_id" : "bench", "count" : 1789 }
{ "_id" : "parking", "count" : 1458 }
{ "_id" : "fast_food", "count" : 1414 }
{ "_id" : "telephone", "count" : 1248 }
{ "_id" : "pub", "count" : 1003 }
{ "_id" : "bicycle_parking", "count" : 922 }
{ "_id" : "grit_bin", "count" : 753 }
{ "_id" : "restaurant", "count" : 720 }
{ "_id" : "cafe", "count" : 637 }
```

- **Top 5 bank branches**

```
> db.birm.aggregate([{"$match":{"name":{"$exists":1}, "amenity": "bank"}},
                     {"$group":{"_id":"$name","count":{"$sum":1}}},
                     {"$sort":{"count":-1}}, {"$limit":5}])

{ "_id" : "Barclays", "count" : 35 }
{ "_id" : "HSBC", "count" : 24 }
{ "_id" : "Santander", "count" : 21 }
{ "_id" : "Halifax", "count" : 18 }
{ "_id" : "NatWest", "count" : 15 }
```

- **Top 10 restaurant trends**

```
> db.birm.aggregate([{"$match":{"amenity":{"$exists":1},"cuisine":{"$exists":1},
                                "amenity":"restaurant"}},
                     {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
                     {"$sort":{"count":-1}}, {"$limit":10}])

{ "_id" : "indian", "count" : 86 }
{ "_id" : "chinese", "count" : 33 }
{ "_id" : "italian", "count" : 26 }
{ "_id" : "french", "count" : 8 }
{ "_id" : "thai", "count" : 7 }
{ "_id" : "american", "count" : 7 }
{ "_id" : "regional", "count" : 6 }
{ "_id" : "pizza", "count" : 5 }
{ "_id" : "mexican", "count" : 5 }
{ "_id" : "fish_and_chips", "count" : 5 }
```

- **Top 10 fast-food trends**

```
> db.birm.aggregate([{"$match":{"amenity":{"$exists":1},"cuisine":{"$exists":1},
                                "amenity":"fast_food"}},
                     {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
                     {"$sort":{"count":-1}}, {"$limit":10}])

{ "_id" : "fish_and_chips", "count" : 177 }
{ "_id" : "chinese", "count" : 137 }
{ "_id" : "pizza", "count" : 57 }
{ "_id" : "sandwich", "count" : 43 }
{ "_id" : "indian", "count" : 42 }
{ "_id" : "burger", "count" : 29 }
{ "_id" : "chicken", "count" : 23 }
{ "_id" : "italian", "count" : 7 }
{ "_id" : "caribbean", "count" : 6 }
{ "_id" : "kebab", "count" : 6 }
```

- **Top 10 leisure trends**

```
> db.birm.aggregate([{"$match":{"leisure":{"$exists":1}}},
                     {"$group":{"_id":"$leisure", "count":{"$sum":1}}},
                     {"$sort":{"count":-1}}, {"$limit":10}])

{ "_id" : "park", "count" : 536 }
{ "_id" : "playground", "count" : 273 }
{ "_id" : "pitch", "count" : 181 }
{ "_id" : "sports_centre", "count" : 116 }
{ "_id" : "picnic_table", "count" : 91 }
{ "_id" : "nature_reserve", "count" : 54 }
{ "_id" : "golf_course", "count" : 49 }
{ "_id" : "fitness_centre", "count" : 38 }
{ "_id" : "garden", "count" : 37 }
{ "_id" : "common", "count" : 33 }
```

- **Top 10 fast-food brands**

```
> db.birm.aggregate([{"$match":{"name":{"$exists":1},"amenity":"fast_food"}},
                     {"$group":{"_id":"$name","count":{"$sum":1}}},
                     {"$sort":{"count":-1}}, {"$limit":10}])

{ "_id" : "Subway", "count" : 67 }
{ "_id" : "McDonald's", "count" : 27 }
{ "_id" : "Greggs", "count" : 25 }
{ "_id" : "KFC", "count" : 24 }
{ "_id" : "Burger King", "count" : 17 }
{ "_id" : "Domino's Pizza", "count" : 11 }
{ "_id" : "Dixy Chicken", "count" : 9 }
{ "_id" : "Pizza Hut", "count" : 8 }
{ "_id" : "Fish & Chips", "count" : 8 }
{ "_id" : "Papa John's", "count" : 7 }
```

- **Top 5 shop trends**

```
> db.birm.aggregate([{"$match":{"shop":{"$exists":1,"$ne":"yes"}}},
                     {"$group":{"_id":"$shop", "count":{"$sum":1}}},
                     {"$sort":{"count":-1}}, {"$limit":5}])

{ "_id" : "hairdresser", "count" : 812 }
{ "_id" : "convenience", "count" : 742 }
{ "_id" : "clothes", "count" : 731 }
{ "_id" : "supermarket", "count" : 282 }
{ "_id" : "car_repair", "count" : 239 }
```

- **Top 5 supermarket brands**

```
> db.birm.aggregate([{"$match":{"name":{"$exists":1},"shop": "supermarket"}},
                     {"$group":{"_id":"$name","count":{"$sum":1}}},
                     {"$sort":{"count":-1}}, {"$limit":5}])

{ "_id" : "The Co-operative Food", "count" : 32 }
{ "_id" : "Tesco Express", "count" : 14 }
{ "_id" : "Spar", "count" : 13 }
{ "_id" : "Costcutter", "count" : 13 }
{ "_id" : "Aldi", "count" : 12 }
```

- **Accessibility for disabled**

```
> db.birm.aggregate([{"$match":{"wheelchair":{"$exists":1},"type": "way"}},
                     {"$group":{"_id":"$wheelchair","count":{"$sum":1}}},
                     {"$sort":{"count":-1}}])

{ "_id" : "yes", "count" : 112 }
{ "_id" : "no", "count" : 7 }
{ "_id" : "limited", "count" : 4 }
{ "_id" : "designated", "count" : 4 }
```

- **Religion of worship**

```
> db.birm.aggregate([{"$match":{"religion":{"$exists":1},"type": "node"}},
                     {"$group":{"_id":"$religion","count":{"$sum":1}}},
                     {"$sort":{"count":-1}}])

{ "_id" : "christian", "count" : 223 }
{ "_id" : "muslim", "count" : 18 }
{ "_id" : "hindu", "count" : 3 }
{ "_id" : "sikh", "count" : 1 }
{ "_id" : "jewish", "count" : 1 }
{ "_id" : "multifaith", "count" : 1 }
{ "_id" : "spiritualist", "count" : 1 }
```

Conclusion

OpenStreetMap data presents an ideal opportunity for me to practice data w rangling and a special w ay to explore citys. Although this review of data is cursory, I think is has been w ell cleaned for the purposes of the exercise.

References

- Introduction to w orking w ith MongoDB and PyMongo
- Data Analysis_Udacity
- OpenStreetMap_Birmingham
- MapData_Birmingham
- Gentle Introduction to MongoDB using Pymongo_Alberto Negron
- MongoDB Manual
- Question About MongoDB_stackoverflow.com