OpenStreetMap Sample Project
Data Wrangling with MongoDB

Yuan Shi

Map Area: Shanghai, China
https://s3.amazonaws.com/metro-extracts.mapzen.com/shanghai_china.osm.bz2
I choose this area because I come from Shanghai, China. I am interested in this area and I want to explore more things about Shanghai.

1. Problems Encountered in the Map

After initially downloading a small sample size of the Shanghai, China area and audit the street name. I noticed that some are named by abbreviations and some are named by Chinese phonetic alphabet. So I used the following regular expressions to rename street abbreviations at the end of the street name.

```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
```

I mapped the street abbreviations into 'proper' unabbreviated names.

```
mapping = { "St": "Street",
 "St.": "Street",
 "Ave": "Avenue",
…
}
```

And then I defined function update_name to rename the street name.

```
def update_name(name, mapping):
    m = street_type_re.search(name)
    other_street_types = [ ]
    if m:
        street_type= m.group( )
    if street_type in mapping.keys( ):
        name = re.sub(street_type,mapping[street_type],name)
    else:
        other_street_types.append(street_type)
    return name
```

I digest some of the final results of fixing the street name as follows:

```
Yongfu Rd. => Yongfu Road
Changhua Rd. => Changhua Road
Bibo Rd. => Bibo Road

Haigang Ave. => Haigang Avenue
Wuchang Ave. => Wuchang Avenue

fumin lu => fumin Road
yongfu lu => yongfu Road
wukang lu => wukang Road
```

I found out that some house numbers were messy. With cases such as house numbers with '-', house numbers with ';', house numbers with space. Using the following code, I updated the house number into a proper house number.

```
def update_number(housenumber):
    if '-' in housenumber:
        convert = housenumber.split('-')[0]
        return convert
    elif ';' in housenumber:
        convert = housenumber.split(';')[0]
        return convert
    elif ' ' in housenumber:
        convert = housenumber.split()[0]
        return convert
    else:
        return housenumber
```

The final results of printing the house number are as follows.
1705-1707 => 1705
1706-1708 => 1706
427-429 => 427
279-1边门 => 279
30-4临 => 30
204;203 => 204
31;33 => 31
1　　号 => 1
112后门 => 112
112-116 => 112
112号103室 => 112103
567 晶采世纪大厦15层 => 567
567号 => 567
3300号环球港3楼 => 33003

I also audited the post code. Most post codes are regular and only few records need to be fixed. I updated the post code using the following code:

```
hanzi=re.compile(u'[\u4e00-\u9fa5]')
def update_postcode(postcode):
    if len(postcode) > 6:
        return postcode[:6]
    elif re.findall(ur"[\u4e00-\u9fa5]",postcode.decode('utf-8')):
        convert = hanzi.sub("", housenumber.decode('utf-8'))
        return convert
    else:
        return postcode
```

The final results of printing the post code are as follows.
```
201315 上海 => 201315
2000080 => 200008
```

2. Data Overview

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

File sizes

dallas_texas.osm	63.9 MB
dallas_texas_osm.json	93.8 MB

```
# Number of documents
> db.osm_data.find().count()
334052
# Number of nodes
> db.osm_data.find({"type":"node"}).count()
297858
# Number of ways
> db.osm_data.find({"type":"way"}).count()
36162
# Number of unique users
> db.osm_data.distinct("created.user").length
1263
#Top 10 contributing user
top_10_contributor = coll.aggregate([{'$group':{'_id':'$created.user',
                                        'count':{'$sum':1}}},
                                    {'$sort':{'count':-1}},
                                    {'$limit':10}])

"{u'_id': u'Chen Jia', u'count': 52675}",
 "{u'_id': u'aighes', u'count': 18555}",
 "{u'_id': u'katpatuka', u'count': 13479}",
 "{u'_id': u'XBear', u'count': 12720}",
 "{u'_id': u'yangfl', u'count': 11303}",
 "{u'_id': u'Holywindon', u'count': 10270}",
 "{u'_id': u'dkt', u'count': 10220}",
 "{u'_id': u'u_kubota', u'count': 9398}",
 "{u'_id': u'jamesks', u'count': 8529}",
 "{u'_id': u'zzcolin', u'count': 8364}"
```

3. Additional Ideas

Additional data exploration using MongoDB queries.

```
#Top 10 amenity
 top_10_amenity = coll.aggregate([{'$match':{'amenity':{'$exists':1}}},
                                  {'$group':{'_id':'$amenity',
                                          'count':{'$sum':1}}},
                                 {'$sort':{'count':-1}},
                                 {'$limit':10}])

"{u'_id': u'bicycle_rental', u'count': 238}",
 "{u'_id': u'restaurant', u'count': 132}",
 "{u'_id': u'parking', u'count': 126}",
 "{u'_id': u'school', u'count': 122}",
 "{u'_id': u'bank', u'count': 57}",
 "{u'_id': u'toilets', u'count': 40}",
 "{u'_id': u'fuel', u'count': 33}",
 "{u'_id': u'hospital', u'count': 32}",
 "{u'_id': u'fast_food', u'count': 31}",
 "{u'_id': u'cafe', u'count': 30}"

#Top 5 bank
 top_5_bank = coll.aggregate([{'$match':{'amenity':'bank','name.en':{'$exists':1}}},
                                  {'$group':{'_id':'$name.en',
                                          'count':{'$sum':1}}},
                                 {'$sort':{'count':-1}},
                                 {'$limit':5}])
```

```
"{u'_id': u'China Merchants Bank', u'count': 2}",
 "{u'_id': u'ICBC', u'count': 2}",
 "{u'_id': u'Bank of China', u'count': 1}",
 "{u'_id': u'BANK OF NANJING', u'count': 1}",
 "{u'_id': u'CCB', u'count': 1}”

#Top 10 highway
top_10_highway = coll.aggregate([{'$match':{'highway':{'$exists':1}}},
                                 {'$group':{'_id':'$highway',
                                        'count':{'$sum':1}}},
                                 {'$sort':{'count':-1}},
                                 {‘$limit':10}])

"{u'_id': u'residential', u'count': 4204}",
 "{u'_id': u'tertiary', u'count': 2831}",
 "{u'_id': u'service', u'count': 2437}",
 "{u'_id': u'unclassified', u'count': 2264}",
 "{u'_id': u'secondary', u'count': 2156}",
 "{u'_id': u'primary', u'count': 1551}",
 "{u'_id': u'motorway', u'count': 1271}",
 "{u'_id': u'motorway_link', u'count': 798}",
 "{u'_id': u'footway', u'count': 748}",
 "{u'_id': u'bus_stop', u'count': 576}"
```

4. Suggestions

After review the dataset, I find that the dataset is incomplete. For example, the number of restaurants in Shanghai is far more than 132. Some of the data even comes from Hangzhou. Although Hangzhou is close to Shanghai, these data should not appear in the dataset of Shanghai area. While I could exclude some of the data which comes from Hangzhou and import some data about restaurant, it’s still hard for me to fix all incorrect data. I think if the data could be more accurate and complete, the results would be more persuasive.

I also explored the top 10 contributors who make the most records. These users are much more active than others. These users could be computers, smart devices or loyalty users. It is reasonable to believe that the data created by these users are more accurate and well formatted than other users’ data. For the users who only contribute couples of points, they may make mistakes as they might be unfamiliar with this city or the platform. We could use data created by these active users and ignore the data created by inactive users. But this may give a blow on other users’ enthusiasm. These users might not be willing to make contributions to the dataset. Some useful and correct data created by these users will lost, which will reduce the correctness of the data. OpenStreetMap is contributed by multiple users, it is quite likely that different people are using different format in reporting the data. If there is a standard data format for reference and all users could abide by the rules, it is entirely possible to import cleaned data to analyze. But in reality it is hard for everyone to obey the rules and typos are inevitable.

Besides, for the reported latitude and longitude, different users may use different devices, such as smart phone GPS, Garmin GPS and so on. Different GPS devices may have different resolution and it is possible that there are two points on the map when it is actually one in reality. We could use third party API to find the wrong information in the dataset and automatically fill in the missing data such as postal code, area, etc. This can further improve the consistency and completeness of the dataset. But there still exists some risks that the the data from third party API is not correct. What’s more, it is possible that the field type of third party API may not match the current dataset.