OpenStreetMap Project

--Data Wrangling with MongoDB

1. Map Area:

Map Area: Vancouver, BC, Canada

https://www.openstreetmap.org/relation/1852574

https://mapzen.com/data/metro-extracts/#vancouver-canada

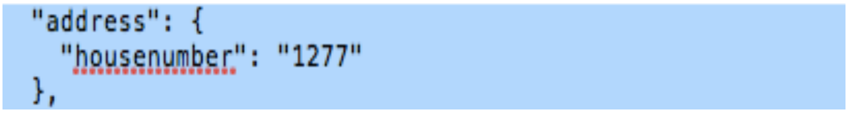
I relocated to Vancouver, Canada about ten years ago, Vancouver is my second home town, I am interested to study the map dataset from that area and see If there is any opportunity to improve the map on OpenStreetMap.org

2. Problems Encountered in the Map

After downloading OSM dataset of the Vancouver area, and I used the code that course provided to take a systematic sample of elements from my original OSM region, then running audit_map.py to audit the sample dataset, I noticed three main problems with the data, which I will discuss in the following order:

- Inconsistent Postal Codes with/without space in the middle (“V7Y1E8”, "V7J 2C1").
- Inconsistent Phone Number ("+1 604 697 9007", "604 709 0907", "1-604-736-8020", "604.700.8708").
- There are many kinds of problems in the value attribute of “address:street”:
    - Abbreviated Street Names (“West 41st **Ave**”)
    - Misspell Street Name (“West Hastings **Steet**”)
    - Incomplete Street Name (“East 2nd”)
    - Street Name with number behind (Expo **Blvd, #3305**)
    - Street Name with city name behind (“Howe **St. Vancouver**”)

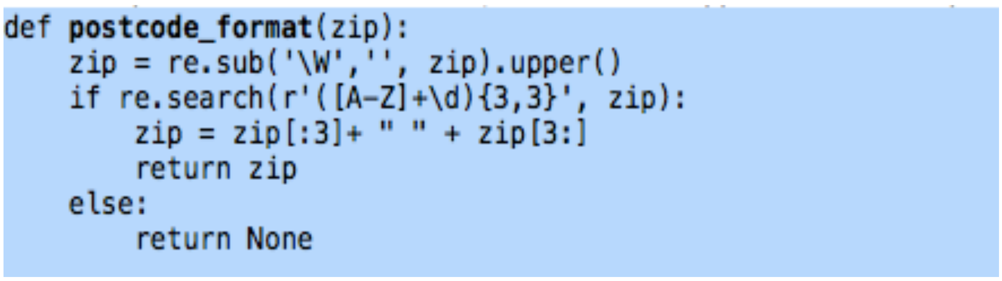
- Invalid Address: the dictionary of address has “housenumber” only.



```
"address": {
  "housenumber": "1277"
},
```

2.1 Postal Codes

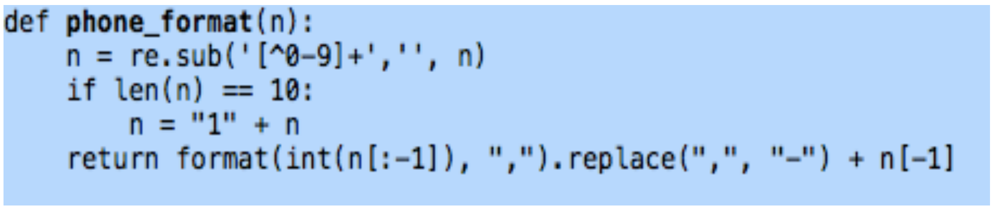
A Canadian postal code is a six-character string with alphanumeric which is in the format A1A 1A1, where A is a letter and 1 is a digit, with a space separating the third and fourth characters. In the dataset, the Postal Codes are inconsistent, there are either with space or without space in the third and fourth characters, I use following function to check if they are valid postal code and then standardize them with upper-case letters and a space separating the third and fourth characters:



```
def postcode_format(zip):
    zip = re.sub('\W','', zip).upper()
    if re.search(r'([A-Z]+\d){3,3}', zip):
        zip = zip[:3]+ " " + zip[3:]
        return zip
    else:
        return None
```

2.2 Phone Numbers

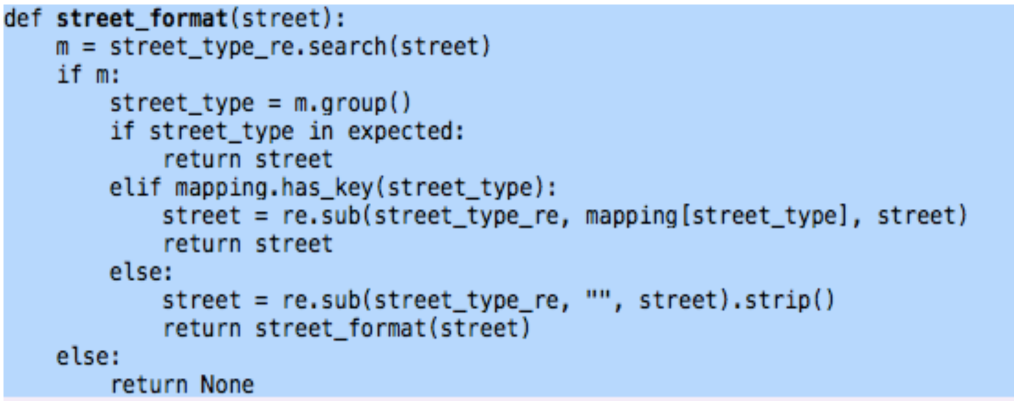
Inconsistent Phone Number ("+1 604 697 9007", "604 709 0907", "1-604-736-8020", "604.700.8708"), I use following function to standardize them to “1-604-123-4567”:



```
def phone_format(n):
    n = re.sub('[^0-9]+','', n)
    if len(n) == 10:
        n = "1" + n
    return format(int(n[:-1]), ",").replace(",", "-") + n[-1]
```

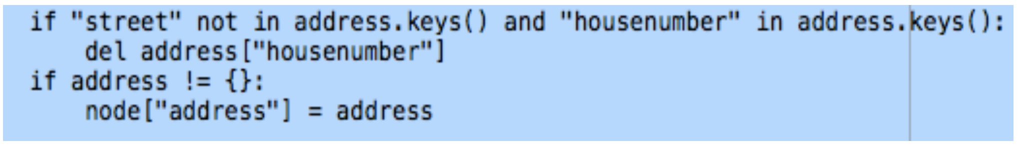
2.3 Addresses

Regarding the problems in the second level of value attribute of "addr:xxxx", I ran following code to fix the abbreviated, misspelled, incomplete street names and street name followed by number or city name:



```python
def street_format(street):
    m = street_type_re.search(street)
    if m:
        street_type = m.group()
        if street_type in expected:
            return street
        elif mapping.has_key(street_type):
            street = re.sub(street_type_re, mapping[street_type], street)
            return street
        else:
            street = re.sub(street_type_re, "", street).strip()
            return street_format(street)
    else:
        return None
```

If the dictionary of address has "housenumber" key only, then "housenumber" key will be deleted:

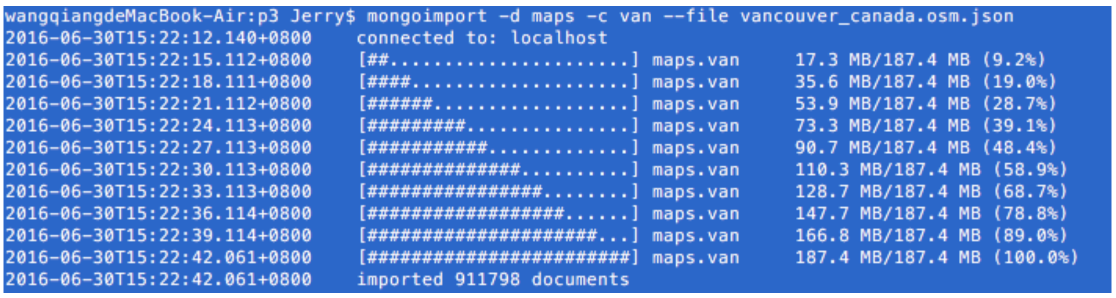


```python
 if "street" not in address.keys() and "housenumber" in address.keys():
     del address["housenumber"]
 if address != {}:
     node["address"] = address
```

3. Overview of the data

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

3.1 Import json file into mongoDB database:



```
wangqiangdeMacBook-Air:p3 Jerry$ mongoimport -d maps -c van --file vancouver_canada.osm.json
2016-06-30T15:22:12.140+0800    connected to: localhost
2016-06-30T15:22:15.112+0800    [##......................] maps.van     17.3 MB/187.4 MB (9.2%)
2016-06-30T15:22:18.111+0800    [####....................] maps.van     35.6 MB/187.4 MB (19.0%)
2016-06-30T15:22:21.112+0800    [######..................] maps.van     53.9 MB/187.4 MB (28.7%)
2016-06-30T15:22:24.113+0800    [#########...............] maps.van     73.3 MB/187.4 MB (39.1%)
2016-06-30T15:22:27.113+0800    [###########.............] maps.van     90.7 MB/187.4 MB (48.4%)
2016-06-30T15:22:30.113+0800    [##############..........] maps.van     110.3 MB/187.4 MB (58.9%)
2016-06-30T15:22:33.113+0800    [################........] maps.van     128.7 MB/187.4 MB (68.7%)
2016-06-30T15:22:36.114+0800    [##################......] maps.van     147.7 MB/187.4 MB (78.8%)
2016-06-30T15:22:39.114+0800    [#####################...] maps.van     166.8 MB/187.4 MB (89.0%)
2016-06-30T15:22:42.061+0800    [########################] maps.van     187.4 MB/187.4 MB (100.0%)
2016-06-30T15:22:42.061+0800    imported 911798 documents
```

3.2 File sizes:

vancouver_canada.osm .... 172.8 MB

vancouver_canada.osm.json ..... 196.5 MB

3.3 Overview

- Number of documents

  ```
  > db.van.find().count()
     output: 911798
  ```

- Number of nodes

  ```
  > db.van.find({"type":"node"}).count()
     output: 761595
  ```

- Number of ways

  ```
  > db.van.find({"type":"way"}).count()
     output: 149393
  ```

- Number of relations

  ```
  > db.van.find({"type":"relation"}).count()
     output: 810
  ```

- Number of unique users

  ```
  > db.van.distinct("created.user").length
     output: 666
  ```

- Top three contribution users

  ```
  > db.van.aggregate([{"$group": {"_id": "$created.user",
                                  "count": {"$sum": 1}}},
                    {"$sort": {"count": -1}},
                    {"$limit": 3}])
     output:
        { "_id" : "keithonearth", "count" : 291939 }
        { "_id" : "michael_moovelmaps", "count" : 113282 }
        { "_id" : "still-a-worm", "count" : 97997 }
  ```

4. Other ideas about the datasets

4.1 Additional data exploration

- **Number of nodes with amenity attribute exist**

```
> db.van.aggregate([{"$match":{"type": "node",
                              "amenity": {"$exists":1}}},
                    {"$group": {"_id": "$type",
                     "count": {"$sum": 1}}}])
                              or
> db.van.find({"amenity":{"$ne": null},"type":"node"}).count()
   output: { "_id" : "node", "count" : 3341 } or 3341
```

- **Top 5 Amenities**

```
> db.van.aggregate([{"$match":{"amenity": {"$exists":1}}},
            {"$group": {"_id": "$amenity",
                    "count": {"$sum": 1}}},
            {"$sort": {"count": -1}},
            {"$limit": 5}])
   output:
        { "_id" : "parking", "count" : 1085 }
        { "_id" : "bench", "count" : 665 }
        { "_id" : "restaurant", "count" : 641 }
        { "_id" : "cafe", "count" : 361 }
        { "_id" : "fast_food", "count" : 291 }
```

- **Top 5 popular cuisines**

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

```
output:
    { "_id" : null, "count" : 308 }
    { "_id" : "japanese", "count" : 45 }
    { "_id" : "chinese", "count" : 42 }
    { "_id" : "sushi", "count" : 30 }
    { "_id" : "pizza", "count" : 16 }
```

- **Maximum speed limit**

```
> db.van.aggregate([{"$match":{"maxspeed": {"$exists":1}}},
                    {"$project": {"_id": "$name",
                                  "maxspeed": "$maxspeed"}},
                    {"$sort": {"maxspeed": -1}},
                    {"$limit":2}])
output:
    { "_id" : "Knight Street", "maxspeed" : "80" }
    { "_id" : "Trans-Canada Highway", "maxspeed" : "80" }
```

Notes: the highway maximum speed limit is 80km/h only in the city area; Canada is a country that pays great attention to traffic safety

4.2 Additional statistics analysis

1) I got total 911798 documents in this database, which were contributed by 666 users, and 55% of them were contributed by top 3 users ("keithonearth", “michael_moovelmaps”, "still-a-worm"), The contributions of users seems incredibly positive skewed.

2) I think “amenity” attribute is a very important and useful field in the map dataset, which is very convenience for users to find different place classified by amenity. However, in this map dataset, the rate of the node with “amenity” field is only 0.43% (3341 vs. 761595).

3) The total restaurants in this dataset are 641, 48% of them didn’t have “cuisine” field, that’s why I’ve got the most popular cuisines is “null”.

4.3 Additional Ideas

To improve the OpenStreetMap dataset, according the statistics analysis above, my opinions:

1) Establishing more close cooperative relations with other websites, mobile apps in the fields of maps, navigation and so on. Let them participated in improving the OpenStreetMap open source data.

2) Let more people or users know this OpenStreetMap.org website by Facebook, twitter and third-part links, specially the business units, where they can upload their business information to the map so that it will make more customers to find them.

3) I suggest some fields in the dataset should be set to mandatory, like 'amenity', 'address', 'phone' and so on, and also establishing format standard guidelines for different area, so that users know what is the right format to upload the data. Setting some fields for mandatory could lower the number of distribution at beginning; However, I believed it would bring more positive impact for the long-term view, which will save a mount of time on data wrangling and cleaning.

5. Conclusion

After the review of the data it's obvious that the Vancouver area is incomplete, I believe it has been well cleaned for the purposes of this project, and I spent 70-80% of total time to deal with data auditing and cleaning on this project. The work efficiency would be greatly improved if the dataset were cleaned and accurate at the beginning. Next, I would like to upload my cleaned data to OpenStreetMap.org if it is possible.

6.References

1) MongoDB Documentation: https://docs.mongodb.com/
2) Udacity Data Wrangling course
3) Udacity forum: discussions.udacity.com