OpenStreetMap Sample Project – Data Wrangling with MongoDB

Chester Fung

Map Area: Hong Kong, China

**1. Problems Encountered in the Map**

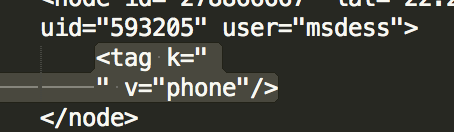
After initially downloading a small sample size of the Hong Kong, China area and running it against a provisional data.py file, the following problems were discovered:

* Unicode-char – initially there were couple of instances where my python file cannot process certain Unicode char. i.e.

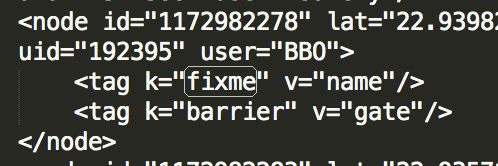




* 4 versions of names in different locales:
  + <tag k="name:en" v="Mid-levels"/>
  + <tag k="name:vi" v="Bán Sơn"/>
  + <tag k="name:zh" v="半山區"/>
* empty k value:



* appears to be incomplete data, “fixme”



Once the data was imported to MongoDB, the following inconsistencies were also found:

**Names Format**

Names were displayed in different formats, including English only names, English and Chinese names, Names with parenthesis









**Cities Names**

**I.** Due to the inconsistencies of the names format, there are many duplicates of the cities names. i.e. Hong Kong







These should all be representing the same location

**II. Also, there are cities which are not part of Hong Kong**, i.e.







**III**. There are also names which appear to be **invalid**



# Sort cities by count, descending

> db.project2\_test1.aggregate([{"$match":{"address.city":{"$exists":1}}}, {"$group":{"\_id":"$address.city", "count":{"$sum":1}}}, {"$sort":{"count":-1}}])

Here are the top two results, beginning with the highest count:



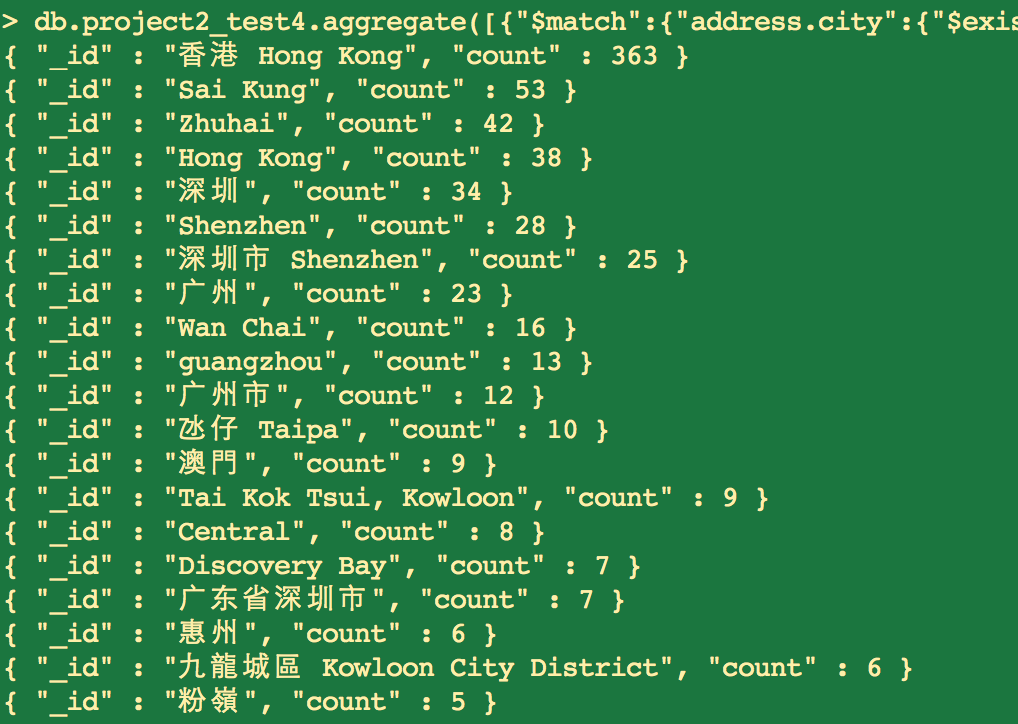
**IV. Cleaning Solutions:**

The following has been performed to fix the problem of having duplicate city names.

There are multiple ways to fix the issue of duplicate city names, one of them is having a dictionary which contains all the possible variations of the city names. I’ve implemented the following in the code to check for 3 city names:



**Before cleaning the data**:



**After cleaning the data**:



**2. Data Overview**

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

File sizes

Hong-kong\_china.osm ......... 449.7 MB

Hong-kong\_china.osm.json .... 641.3 MB

# Number of documents

> db.project2\_test1.find().count()

2374154

# Number of nodes

> db.project2\_test1.find({"type":"node"}).count()

2161681

# Number of ways

db.project2\_test1.find({"type":"way"}).count()

212473

# Number of unique users

db.project2\_test1.distinct('created.user').length

1259

# Top 1 contributing user

db.project2\_test1.aggregate([{"$group":{"\_id":"$created.user", "count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":1}])

{ "\_id" : "hlaw", "count" : 515440 }

# Number of users appearing only once (having 1 post)

db.project2\_test1.aggregate([{"$group":{"\_id":"$created.user", "count":{"$sum":1}}}, {"$group":{"\_id":"$count", "num\_users":{"$sum":1}}}, {"$sort":{"\_id":1}}, {"$limit":1}])

{ "\_id" : 1, "num\_users" : 231 }

**3. Additional Ideas**

One major hurdle in cleaning the data of my area is that my dataset contains unicode characters. Translating city names across different languages (in my case English and Chinese) is a difficult task. After discussing this in the forum, here are a few ideas on how to tackle this problem:

1. transliterate Unicode char into string

2. predefine dictionaries which contains keys and values of variations of city names

3. check if two cities overlap by comparing latitude/longitude

I chose to use number 2 to clean city names in my dataset. The difference was shown in the screenshots above.

Programmatically number 2 was not very difficult, as python has good support of Unicode characters operations. However, the downside of using this approach is that many city names will need to be defined and fine-tuned as we go thru all the different variations of the particular city names

**Additional data exploration using MongoDB queries**

# Top 10 appearing amenities

db.project2\_test1.aggregate([{"$match":{"amenity":{"$exists":1}}}, {"$group":{"\_id":"$amenity", "count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":10}] )

{ "\_id" : "parking", "count" : 1615 }

{ "\_id" : "school", "count" : 1382 }

{ "\_id" : "toilets", "count" : 1028 }

{ "\_id" : "restaurant", "count" : 653 }

{ "\_id" : "bus\_station", "count" : 635 }

{ "\_id" : "shelter", "count" : 619 }

{ "\_id" : "bank", "count" : 432 }

{ "\_id" : "place\_of\_worship", "count" : 420 }

{ "\_id" : "fuel", "count" : 286 }

# Biggest religion

The most number of religion is a bit surprising…given majority of population in HK is Chinese

db.project2\_test2.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"place\_of\_worship"}},{"$group":{"\_id":"$religion", "count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":1}])

{ "\_id" : "christian", "count" : 140 }

# Most popular cuisines

> db.project2\_test2.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"restaurant"}}, {"$group":{"\_id":"$cuisine", "count":{"$sum":1}}},{"$sort":{"count":-1}}, {"$limit":2}])

{ "\_id" : null, "count" : 365 }

{ "\_id" : "chinese", "count" : 136 }

**Conclusion**

There were a few surprises as I thought I was familiar with the area of Hong Kong. Though there were challenges and deficiencies as mentioned earlier in the report, overall it was a good exercise to learn to clean and wrangle the data