Open Street Map Project

Eric Huang

1. Problems Encountered in the map

The map data I used is downloaded from the following website:

https://mapzen.com/data/metro-extracts/metro/shanghai_china/

I choose Raw Open Street Datasets and click the "OSM XML" version. I went through three steps to find problem and automatically clean the dataset:

1) Map parse => Get a whole picture of data and decide for future cleaning targets;
2) Audit data => Determine targets and go more in detail how to clean data;
3) Format data => Use real data and make cleaning process from xml to json automatically;

1.a Map Parse

Firstly I tried to use python (mapparser.py) to figure out what distinct keys it contains. If possible to import big dataset into local mongodb database, map-reduce function of mongo shell can also be used to get keys and appear times in database.

```
> mr = db.runCommand({
  "mapreduce" : "shanghai",
```

```
    "map" : function() {
       for (var key in this) { emit(key, 1); }
    },
    "reduce" : function(key, value) { return Array.sum(count); },
    "out": "shanghai_keys_count"
})
> db.shanghai_keys_count.find().pretty()
```

I got some interesting key names from the command output. The one that caught my eyes most is that some of fields such as “name” have a lot of languages formats such as “name:en” and “name:fr”. And also some potential fields – addr:postcode, addr:street, contact:phone which will be easily result in human mistakes but easy to find out and fix by program. Then, I go to next step to go into detail of these fields.

1.b Audit Data

I will go through previous four parts one by one.

The first part I will go through in auditing is “addr:street”. When going through some unformatted street name, I also find that “addr:city” has problem. There exists data not in Shanghai. Now my cleaning target becomes clear:

1) Discard data not in Shanghai, i.e. “addr:city” is not Shanghai related (both in English and Chinese);

2) Audit street name (only in English since I have no background of how Chinese road name should look like) to proper format;

Then I go though checking postcode. And I find that some postcodes do not meet specification of continuously 6 digits. Since I cannot recover these invalid postcodes, I have to discard postcode fields of these data when I generate json format of data.

Finally, I go through phone number and also find out that different formats exist in the dataset. However, unlike postcodes, they are recoverable. What I need to do is extracting useful digits out of original ill formatted phone number and recombining them together into a unified formatted one. For phone number, I use format of only 11 digits number without “ ” and “-” and without country code since I already know

that Shanghai is in China (+86).

Note that audit.py will output all city names (Chinese characters are in the right form.) that are not in Shanghai and then sequentially output ill-formatted street name => formatted street name, ill-formatted phone number => formatted phone number, list all not valid postcode and finally list all not in Shanghai records' street address info (all Chinese characters are in the Unicode form).

1.c Format Data

After implementing the experimental python codes to clean row data, it is time to really clean row data and output in json format. For detail, please see comments of data.py.

Through first running older version of this script, I found that many data in dataset are not informative, i.e. they only contains mandatory fields. As a result, in the submitted version of data.py, I added a filter to drop all data with only mandatory fields. For mandatory fields, I mean "pos", "_id", "type", "id", "created", "created_by". And although for many data, "amenity" field is valid and human understandable, they are not properly categorized. I got the following ill formatted results:

"pub" vs "pub;ktv"

"Centre commercial" vs "public_building"

"car_parking" vs "bicycle_parking" vs "parking"

"进才中学北校" vs "建平世纪中学" vs "联华超市" vs "school"

First is multiple categories problem. The second shows different formats – previous one with first capital letter while later one with "_" and without first letter capitalized. The third one shows overlapping but different amenity category. The fourth one contains both Chinese and English version of similar amenity category. Note that "中学" and "校" in Chinese means "school". Although it is recoverable, they are really hard to fix. For example, is it better to combine "car_parking" and "bicycle_parking" into "parking" or manually decompose "parking" to car and bicycle. I believe it depends on the target of my later analyze but it will not in the scope of this report. As a result, I do nothing related to this ill format problem.

2. Data Overview

This section contains basic statistics of dataset and queries used to gather them.

Data file size:

shanghai_china.osm……633.7 MB

shanghai_china.osm.json……169.1 MB

The following are some queries I made for data overview. You can also find them in my import_mongodb_and_query.py file. If you want to get the detail results for some of queries below, please run python script.

Number of records

> *db.shanghai.find().count()*

532701

Number of unique users

> *db.shanghai.distinct("created.user").length*

1754

Top three most mentioned amenities

```
> db.shanghai.aggregate([
    {"$match":{"amenity":{“$exists”:1}}},
    {"$group":{"_id":"$amenity", "count":{"$sum":1}}},
    {"$sort": {"count": -1}},
    {"$limit": 3}
  ])
{u'count': 2428, u'_id': u'bicycle_rental'},
{u'count': 1254, u'_id': u'parking'},
{u'count': 1144, u'_id': u'restaurant'}
```

Number of nodes

> *db.shanghai.find({“type”:”node”}).count()*

170095

Number of ways

> *db.shanghai.find({“type”:”way”}).count()*

362606

Amenities only appear once (I omit query results here.)

```
> db.shanghai.aggregate([
    {“$group”: {"_id":"$amenity", "count": {“$sum”: 1}}},
    {“$group”: {"_id":"$count", "num_amenity":{"$sum":1}}},
    {“$sort”: {"_id":1}},
    {“$limit”: 1}
  ])
```

Display phone number (I omit query results here.)

```
> db.shanghai.distinct("contact:phone")
```

Display postcode (I omit query results here.)

```
> db.shanghai.distinct("address.postcode")
```

3. Additional Ideas

3.a Data exploration with language problems

The first big problem encountered in analyze is different languages. For example, some “name” fields contains both Chinese and English characters. When using regular expression to filter out valid names, those names consists of Chinese characters may be treated wrongly. As a result, I believe it is a good idea to restrict input to only English characters, and it will result in several advantages:

1) It is much easier to use regular expression in English than with other languages in python, especially when I do not know what kind of languages will appear.

2) Data Analyst will not need to have any prior language requirements such as knowing Chinese before writing or reading python data cleaning program.

3) This action will also avoid self-conflict value detection and cleaning and redundant information in different languages.

One way to work around this problem is to add or overlap existed key such as “name” with English name. This English name can either from other keys such as “name:en” or from Google translation API.

Some anticipated problems included in solving multiple languages problem is:

1) How to automatically detect language types;

2) How to deal with multiple languages in one string;

3) How to encode and decode different languages.

The following is one detailed problem I encountered during data cleaning. All characters in Chinese pinying are valid English characters. “lu” in pinying means road in English. However, some road names also contain “lu”. I believe it is a really hard task to automatically distinguish “lu” in road name and “lu” meaning road while replacing “lu” meaning road by English word “road”.

3.b Amenity group name investigation

I do data cleaning for future analyze. As a result, a good category of what a location’s function is a must and most valuable information that should be affiliated with data. In open street map dataset, I believe it should be “amenity”. During cleaning of open street data, I find that a lot of records do not contain “amenity”. And those with “amenity” key have either overlapping but different categories or different levels. It is a good idea to automatically group “amenity” to some high-level predefined standard groups, then assign back to “amenity” field in data cleaning. Furthermore, for those records without “amenity”, it is probably a good idea to use existed information in data to assign “amenity” to it. For example, if name of a position contains words such as “restaurant”, it can be assigned “eating” tag.

Some anticipated problems included in group and assign amenity are:

1) How to predefine amenity groups to properly meet requirements of later analyze;

2) How to automatically assign a proper amenity to a record whose amenity is undefined without too much human interferance.

3.c Conclusion

During the investigation, I try to clean a big data set. From my point of view, this dataset has two main problems:

1. Different language problems;

2. Overlapping and different level of amenity;

The problems will result in big issues if left to next stage. And I make examples for each of two problems and discuss how to deal with these problems in additional ideas.

Reference

https://docs.mongodb.com/manual/core/map-reduce/

https://docs.mongodb.com/manual/reference/operator/query/regex/

http://www.cnblogs.com/huxi/archive/2010/07/04/1771073.html

https://docs.mongodb.com/manual/reference/program/mongoexport/

https://docs.mongodb.com/manual/reference/program/mongoimport/

https://www.travelchinaguide.com/essential/area_zip/shanghai.htm

https://api.mongodb.com/python/current/