OpenStreetMap Sample Project
Data Wrangling with MongoDB

Darui Zhang

Map Area: Tacoma, WA, United States
Data source: Overpass API:http://overpass-api.de/query_form.html

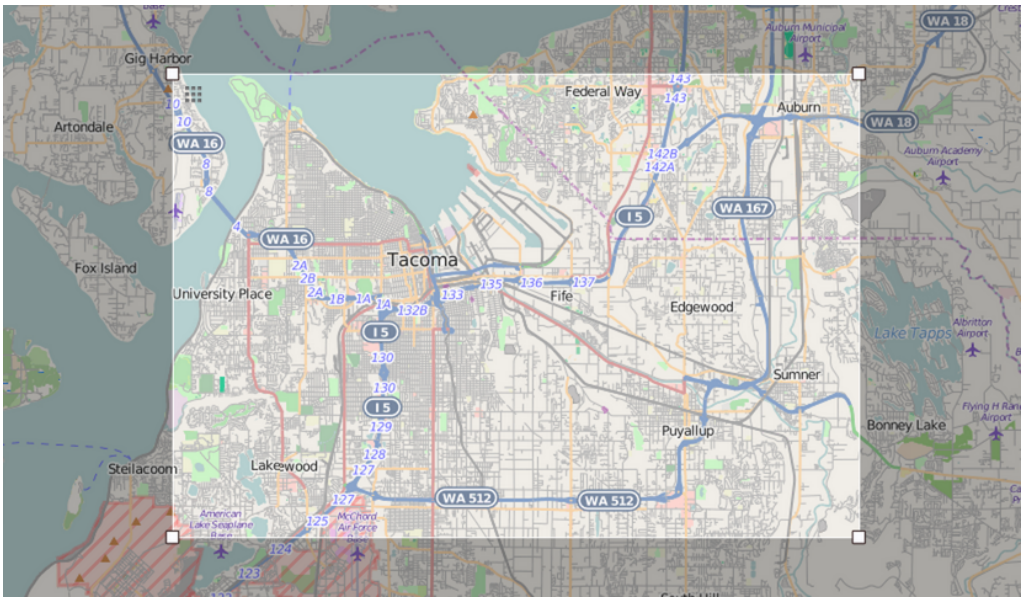

Figure 1. Tacoma, WA. Selected area on OpenStreetMap
(minimum_latitude: 47.1622, minimum_longitude:-122.6404
maximum_latitude: 47.1622, maximum_longitude: -122.1714)

1.Data Wrangling Process

In this project, the data wrangling process was conducted following a cycle of auditing, cleaning and repeat. After the data has been cleaned, an analysis was conducted using MongoDB.

Step1:
source code: "step1_audit. py"
tasks:

- audit/count primary tag
- audit/count secondary tags
- audit street names

I/O: Input original .osm file, no output

Step2:
source code: “step2_parse. py”
tasks:

- updata street names
- parse the osm file into json

I/O: Input original .osm file; output parsed json file.

Step 3:
source code:“step3_clean. py”:
tasks:

- audit and clean secondary tags

I/O: input parsed json file; output cleaned json file

Step 4:
source code: “step4_analysis. py”
tasks:

- import the data into MongoDB
- data analysis using queries in MongoDB

I/O: input cleaned json file, no output

2. Problems Encountered in the Map

From the initial data auditing in step 1, we found 2 challengings in this dataset.

- Irregular street name
- Reductant and necessary secondary tag

These problems and the approaches to solve them will be discussed in detail in the following sections.

2.1 Irregular Street Name

From the data audit in step 1, I found the street names do not come into a uniform naming convention; some using abbreviation; some are purely entering errors.
Our expected street name convention is that: a word representing a street (eg. “street”), “Avenue”,’Boulevard”) is at the end of the street name (eg. “South Puget Sound Avenue”). However, I found some exceptions:

- The word represent street is not at the end of street name

An example is “14th Avenue Southwest”. The word that representing street is the second last word. Since, they are actual correct street names, we do not need to change them. I added an additional list of string represent the directions(eg. “Southwest”, “East”), so that this kind of street name will not be flagged out as irregular street names.

- Highway

Highways have a different naming convention (eg. “State Route 410”). Since they are also correct street name, they do not need to be corrected.

- Abbreviation

We want to correct the abbreviation to its full name so that the data is more consistent. For example, “Ave” will be replaced by "Avenue". We use information found in the data auditing step to create a list for mapping the abbreviated words to full words. The correction is done in Step 2 data parsing, using a function “update_name”.

- More than one world have abbreviation

An example is “Tacoma Ave S”, both the last word and the second last word have abbreviation. So when updating the name, both the last and second last world needs to be checked.

- Data entry error

They are pure errors, for example 'http://local.safeway.com/wa /tacoma-1594.html' is a street name found in the data set. However, it is actual the website of this location. These kind of errors are deleted.

2.2 Irrelevant and Redundant Secondary Tags

In the initial data auditing, I found there is more than 300 secondary tags. Many of them have a very small count. After exploring the data more, I found some of them contains

irrelevant or redundant information. Some tags provide the same information, but they have a different name. The different situations are treated respectively.

- Irrelevant keys / delete

Some of the keys do not provide useful information for common user. For example, some data that comes from 'tiger_import' will have the tag "tiger:reviewed":"yes" and "tiger:upload_uuid":""bulk_upload.pl-d6ec3555-9de6-41da-8fa4-ed8c95434959". This information might be useful in the original data source, but have little value to the user of OpenStreetMap. For this type, since they do not contain much relevant information, they are deleted from the data set.

- Reductant information/ check and delete

Some keys provide redundant information. For example, we have a record contains:

"name": "96th Avenue Court West",

"tiger:name_direction_suffix": "W",

"tiger:name_type": "Ct",

"tiger:name_base": "96th Avenue",

The information provided in the later three tags already contained in the first tag. For this type, we check if the primary tag ("name") exist, if it exist we can delete the secondary tags.

- Inconsistent name /rename

The data from different source may have different keys for the same information, for example, "'tiger:source','tiger:county', are the same as "source" and "county" in other data set. some other data set use 'is_in' to represent "county". For this type, we rename the specially tag name to the more common names.

- Results

After the data cleaning, the number of secondary tag was reduced from 337 to 306, data size was reduced from 83.9-61.0 MB.

3. Data Overview

3.1 File sizes:

tacoma.osm ..... 59.8 MB (original .osm file)
tacoma.json ..... 83.9 MB (parsed .json file)
tacoma_cleaned.json … 61.0 MB (cleaned .json file)

3.2 Data analysis using MongoDB

```
# Number of data records
>db.tacoma.count()
271,894

# Number of ways
> db.tacoma.find({'type':'way'}).count()
28,508

# Number of nodes
> db.tacoma.find({'type':'node'}).count()
243,364

# Number of schools
>tacoma.find({'amenity':schools}).count()
278

# Number of unique users
>pipeline = [ { "$group" : { "_id" : "$created.user","count": {"$sum": 1 }}},
{ "$sort" : { "count" : -1 }},
{ "$skip" : 1 }]
>result = db.tacoma.aggregate(pipeline)
>print(len(result["result"]))
278

# Top 5 contributing users
[{'_id': 'woodpeck_fixbot', 'count': 52087},
 {'_id': 'KTyler', 'count': 29246},
 {'_id': 'STBrenden', 'count': 28343},
 {'_id': 'zephyr', 'count': 9859},
 {'_id': 'compdude', 'count': 9012}]
```

4. Other Ideas About the Dataset

More than 300 secondary tags can be found in this data set. This is partly because the data come from various sources, each has their naming convention. It would be beneficial if we can develop a category scheme which can group these attributes and organize information effectively. For example, in current data set "restaurant" and "fast_food" have the different value in "amrity". However, fast food can be seen as a subset of restaurant. In this way, we can reduce the number of tag and make the data more consistent. In addition, the tags are checked manually. More automatic method would be helpful. For example, we can check the tag name and tag value to decide whether a certain tag belongs to a category.

Additional Data Exploratory using MongoDB

Top 5 data source:

```
[{'_id': 'tiger_import_dch_v0.6_20070830', 'count': 8432},
 {'_id': 'Bing', 'count': 6119},
 {'_id': 'PGS', 'count': 2646},
 {'_id': 'bing', 'count': 1863},
 {'_id': 'yahoo_wms', 'count': 216}]
```

(Bing were counted separately as capital inicial one and non-capital inicial one)

Top 5 common amrity:

```
[{'_id': 'school', 'count': 278},
 {'_id': 'place_of_worship', 'count': 257},
 {'_id': 'restaurant', 'count': 155},
 {'_id': 'parking', 'count': 132},
 {'_id': 'fast_food', 'count': 121}]
```

5. Conclusion

In this project, the geographical data of the Tacoma, WA area from the OpenStreetMap was audited, cleaned and underwent data analysis using MongoDB. The data wrangling process follows the cycle of auditing, cleaning and repeat. Two of the problems encountered in this project are: the irregular street names, and irrelevant or redundant secondary tags. The abbreviated name is replaced with the full name. The irrelevant redundant data are deleted. Inconsistent tags are renamed to more common names. Further improvement can be made in creating more organized category and reduce the number of tags. More automatic approach can also be made to reduce the work of manual data auditing and cleaning in order to convert the tag from different sources to more generalized names.