OpenStreetMap-Case-Study

July 3, 2017

1 About Map Area

1.1 Location: Beijing, China.

Map Url:

- https://mapzen.com/data/metro-extracts/metro/beijing_china/
- https://www.openstreetmap.org/relation/912940

1.2 File Size

The data file is about 181 MB in size (uncompressed).

1.3 Reason to Choose this Area

I choose Beijing as the map area because I had lived in Beijing for several years, so I'm familar with this city and would like to check the quality of the map data from OpenStreetMap.org.

2 Steps

There are five steps of this analysis:

1. Generate the Sample data;
2. Check how many different kinds of tags in the data;
3. Audit potential problems for each tag;
4. Fix the problems;
5. Prepare the data to be inserted to a SQL database.

2.1 Step 1: Generate the Sample data

Execute the follow command in the Shell:

```
python data/gen_sample.py
```

This will output a sample data file in `data/` folder with name `sample.osm`.

2.2 Step 2: Check how Many Different Kinds of Tags in the Data

Although this is not the requirement of this project, I'm a little curious about how many different kinds of tags are there in the data. And I used the original data file instead of the sample data to check this:

```
In [49]: import xml.etree.cElementTree as ET
         from pprint import pprint

         def count_tags(filename):
             tags = {}
             for event, ele in ET.iterparse(filename, events = ('start', 'end')):
                 if event == 'end':
                     if ele.tag in tags:
                         tags[ele.tag] += 1
                     else:
                         tags[ele.tag] = 1
             return sorted(tags.items(), key = lambda x: (-x[1], x[0]))

         tags = count_tags('data/beijing_china.osm')
         pprint(tags)
```

```
[('nd', 1018804),
 ('node', 853320),
 ('tag', 360734),
 ('way', 127592),
 ('member', 61361),
 ('relation', 5657),
 ('bounds', 1),
 ('osm', 1)]
```

The results shows there are totally 8 different kinds of tags in the data, and `nd`, `node`, `tag`, `way` are the most common tags in the data. Because the data that we will extract are mainly in these four kinds of tags, so the audit and clean process are focused on these tags. For `tag` tag, it may be the tag of a `node` tag or a tag of a `way` tag, so we call them as `nodes_tags` and `ways_tags`, seperately. For `nd` tags, we call them `ways_nodes`.

2.3 Step 3: Audit Potential Problems

In order to audit potential problems of this data, I choose to focus on two aspects of the data quality:

- *Data Fields Types*;
- *Data Fields Validity*.

The code for audit process is in the `audit` function within the `audit.py` file, which takes the data path as input and return the audit result as output. The output is a dict, and its structure is shown as below:

```
{'field_types':
    {'node':
        {'id': (type_1, type_2, ..., type_n),
         'lat': (type_1, type_2, ..., type_n),
          ......
         'timestamp': (type_1, type_2, ..., type_n)},
     'way':
        {'id': (type_1, type_2, ..., type_n),
         'user': (type_1, type_2, ..., type_n),
          ......
         'timestamp': (type_1, type_2, ..., type_n)},
     'node_tags':
        {'k': (type_1, type_2, ..., type_n),
         'v': (type_1, type_2, ..., type_n)},
     'way_tags':
        {'k': (type_1, type_2, ..., type_n),
         'v': (type_1, type_2, ..., type_n)},
     'way_nodes':
        {'ref': (type(int()))}
     },
 'field_validity':
     {
        'node': {'lat': ['min', 'max'],
                 'lon': ['min', 'max'],
                 'timestamp': ['min', 'max']},
        'way': {'timestamp': ['min', 'max']},
        'node_tags': {'postcode': ('wrong_value', ...)},
        'way_tags': {'name_en': {'unknown_way_type': ['way_name',......]},
                     'postcode': ('wrong_value', ......)},
        'way_nodes': {}
     }
}
```

Within the `audit(osm_file)` function:

- `update_field_types(e, tag)`: this function will update the set of its field types based on current element for a given tag, so after iterative across the file, we can get the results of the `field_types`;
- `validate_*`: the functions that start with `validate_` will check the validity of some predefined fields based on the type of the current element, so after iterative across the file, we can get the result of `field_validity`.

In addition, there are several helper functions within the `audit()` function to avoid too much repeated code.

2.3.1 Problems Encountered in the Map

1. Over Abbreviated Street Names: e.g. (`"W. Dengshikou Str"`)

2. Inconsistent Street Names: e.g. (`"Yongfeng Lu"`, `"Liangshidian jie"`)

3. Incorrect Postal Codes: e.g. (`k="addr:postcode" v="010-62332281"`)

   Note: Because the map area that I choose is in China, which makes the field `addr:street` is in Chinese, however, this is an english project and there is a field `name:en`, so after checking the field `name:en` and `addr:street`, I found the number of field `name:en` is much more than the number of field `addr:street`, and I believed the field `name:en` of the element `way` gives the information about the name of the location. So I decided to use field `name:en` to audit instead of field `addr:street`.

About the `way` names After the audit process, I noticed that the most common problems are about names, and there are three kinds of the name problems:

1. Over Abbreviated Street Names, such as:

   - `"W. Dengshikou Str"`: `"Str"` should be `"Street"`;
   - `"Lugu Rd."`: `"Rd."` should be `Road`;

2. Inconsistent street names which use Chinese "Pinyin" to represent the English name, such as:

   - `"Yongfeng Lu"`: `"Lu"` should be `Road`;
   - `"Liangshidian jie"`: `"jie"` should be `Street`;

3. Uncommon street names which are difficult to deal with, such as:

   - `'Habor': set(['Solana Blue Habor'])`
   - `'Cheng)': set(['Interwest (Zhu Yu Cheng)'])`
   - `'Middle': set(['North 3rd Ring Road Middle'])`

For the first and second kinds of problems, they can be easily cleaned up programmatically, however, for the third kind of problems, it is hard to cleaned up programmatically. The reason is these street names are not regularly, even the local people can't easily figure out their english name, so these street names have to be dealt with one by one with care.

Therefore, in this case study, I will only focus on the first two kinds of problems and ignore the third.

About the postal codes After running the audit script, I found a obvious error with the postcode in the sample data: `k="addr:postcode" v="010-62332281"`. "010-62332281" is a telephone number and not a postcode. Since it's hard to get the postcode from the telephone number automatically, the proper solution is just delete this wrong postcode (it maybe possible to use geocode api of Google Maps based on the street name to obtain the postcode, but this is beyond the scope of this case study and the result from geocode api still need to be checked, so I just choose the simple way to deal with this postcode problem).

2.4 Step 4: Fix the Problems

For the problems of way names, I use the follow function to fix it:

```
def update_way_names(name, mapping):
    for k, v in mapping.items():
        if k in name:
            name = name.replace(k, mapping[k])
            return name
    return name
```

and for the problems of the postcode, I just ignore the problem filed when converting osm file to csv file in the `data.py` script.

2.5 Step 5: Prepare the data to be inserted to a SQL database

Once `audit.py` and `data.py` are completed, the osm file will be converted to csv file after execute the `data.py` script, and some of the problems will be fixed in the meantime.

After this converting process, it is very easy to import the csv files to a SQL database.

3 Data Overview and Additional Ideas

3.1 File sizes

- beijing_china.osm: 181 MB;
- osm_beijing.db: 167 MB;
- nodes.csv: 67 MB;
- ways.csv: 7.2 MB;
- nodes_tags.csv: 2.9 MB;
- ways_tags.csv: 8.2 MB;
- ways_nodes.csv: 23 MB.

3.2 Number of nodes

```
SELECT COUNT(*) FROM nodes;
```

The result is: 853320

3.3 Number of ways

```
SELECT COUNT(*) FROM ways;
```

The result is: 127592

3.4 Number of unique users

```
SELECT COUNT(DISTINCT(T.uid)) FROM
(SELECT uid FROM nodes UNION ALL
SELECT uid FROM ways) as T;
```

The result is: 1798

3.5 Top 10 contributing users

```
SELECT T.user, COUNT(*) AS num FROM
(SELECT user FROM nodes UNION ALL SELECT user FROM ways) as T
GROUP BY T.user
ORDER BY num DESC
LIMIT 10;
```

The result is:

- Chen Jia|237945
- R438|142732
- hanchao|66853
- ij_|51901
- Алекс Мок|47522
- katpatuka|23521
- m17design|21599
- Esperanza36|18527
- nuklearerWintersturm|16474
- RationalTangle|13748

It seems that the contributions of users is skewed, we can use the followed SQL to compute the overall contributions of the top 10 contributing users:

```
SELECT SUM(NUM.num) FROM
(SELECT T.user, COUNT(*) AS num FROM
    (SELECT user FROM nodes UNION ALL SELECT user FROM ways) as T
    GROUP BY T.user ORDER BY num DESC LIMIT 10) as NUM;
```

The result shows that the overall contributions of the top 10 contributing users is 640822, which makes up about 65.3% of all users.

3.6 Additional Data Exploration

3.6.1 Top 10 Amenities

```
SELECT value, COUNT(*) as num FROM
nodes_tags WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

The result is:

- restaurant|1267
- bank|452
- toilets|359
- fast_food|328
- cafe|274
- school|161
- telephone|151
- bar|142
- parking|135
- atm|114

Well, it's interesting to find that the most popular amenity is restaurant, however, this is not a surprise because this is Beijing. Further, I would like to check the top 10 popular cuisines in Beijing.

3.6.2 Top 10 Popular Cuisines

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags JOIN
(SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') as T
ON nodes_tags.id=T.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 10;
```

The result is:

- chinese|166
- japanese|21
- italian|17

- `pizza;american|15`
- `regional|11`
- `international|10`
- `pizza|9`
- `american|7`
- `asian|7`
- `german|5`

It's very clear that the Chinese restaurant is the most popular one, and the number of Chinese restaurant is far more than the other cuisines.

From the above results, we can also notice that there are repeated cuisines in the data: `pizza;american` vs. `pizza` and `american`, which need to be cleaned up.

3.7 Additional Ideas

3.7.1 OSM Data need a New Data Type: `area`

When I audit the data I found that `ways` not only contains "ways" but also contains "areas": such as building, university, park, etc. In my opinion, we need three kinds of abstract spatial data types to represent the spatial data: point, line, and area. In the OSM map data, there are only "node" and "way" which represents the "point" and "line", seperately. So there is a missing data type: area. The reason to seperate the "way" and "area" is that "way" is more suitable to stand for the "road", "street" and so on, and "area" is more suitable to stand for the "building", "park" and so on. Also, "line" is made up of "point", and "area" is made up of "line", so if we add the third type: "area", then the data structure will be more complete and less confused.

3.7.2 Potential problems/challenges Added for the Second Submission

Besides the benefits of the proposed improvement, there exist some potential problems/challenges to be addressed. The most obvious one is that sometimes it is difficult to decide whether a place should be an "area" or just a "way". For example, in many cities of China, there will be some commercial pedestrian streets (with restaurants and cloth shops, etc.) for shopping, basically we can treat them as "way". However, some commercial pedestrian streets are more like a square than a street, which indicates that it should be treated as "area". Therefore, the user who would like to create the data should consider this problem prudently and then choose a suitable type, which may need some professional knowledge.

The reason for this challenges is that "area" is also made up of multiple points of which the start point is the same as the end point, so "area" is essentially a special type of "line". In addition, I think this maybe the reason that OSM map data only contains points and lines.

However, this should not be the reason for the missing of the "area" type, because "area" and "line" are very different spatial data abstract types and they should have different set of properties. So I would suggest that "area" should be added to the abstract data types of the OSM data.

4 Conclusion

From this projct I learned that data cleaning is a time-consuming process, it is not as cool as the machine learning models, but it's a really important part of the whole Data Science project. Because if the quality of the input data is not good, then the model can not be a good model, because the model is learned from the data. However, sometimes maybe there is no end of the data cleaning process, because if the data is big enough we can not guarantee that there is no error in the data. We can only audit and clean the data thoroughly to make the data better and better until the data quality meet the requirement of the current project.