OpenStreetMap Project

Jina Lee

This project extracted and cleaned OpenStreetMap data and stored it into MongoDB so that users can explore quality data fast. OpenStreetMap is map information in a topological data structure collected by volunteers. It consists of four major elemnts; Node, Ways, Relation and Tags. Nodes present geographic information of specified locations by points and create log such as user, update time etc. Ways represent linear features such as areas, rivers, streets and create log such as user, update time etc. Relations are representing relationship between Nodes and Ways, and Tags contains metadata of its Nodes or Ways elements.

Map Area: New York City, NY, United States

1. Problems Encountered in the Map and

The way to audit data

This project audited the data's quality in five perspectives to assure concrete data analysis result.

- Completeness :
  - Computed descriptive statistics of data: number of elements, Counted number of records for each attribute and missing values.
- Validity
  - Audited if street name, postcode, height data are in appreciate formats.
- Accuracy
  - Checked if postcode starts with New York's code: 0 or 1.
- Consistency:
  - Not applicable here since, this project use data from only one source.
- Uniformity:
  - Checked 'height' if it is in the same unit, and corrected data in different units to foot.

Corrected data

- Postcode is missing and integrated in address field in some data
  : cut the postcode in the street field to postcode field

  ```
  'street': 'West 80th Street NYC 10024'
    → 'street': 'West 80th Street', 'post_code': 'NYC 10024'
  ```
- Fixed over-abbreviated or misspelled street type names
  : 'aveneu' → 'Avenue', 'pkwy'→ 'Parkway', 'hwy' → "Highway"

- Unified unit of height as feet and changed data type into numeric
  : '7ft' → 7, ‘8 feet’ → 8, ’77.7m’ → 254.9212

Designing the shape of data, reflecting attribute types revealed through auditing.

- Filtering out attributes that have non-character name.
- Grouped related attributes
  - created: to data creation information in a dictionary such as user, timestamp, etc.
  - address: grouped attributes under 'Address' such as addr:housenumber, addr:zipcode etc.
  - others which has sub-categories : grouped attributes that its names contains ':', for example, 'gins:housenumber.'
- Combining 'latitude' value and 'longtitude' value into a list 'pos' for using GEO2D index in MongoDB
- Transformed data type of timestamp from string to datetime object.

2. Data Overview

This project used OpenSteetMap New York area of OSM filetype which is downloaded from MapZen.

Basic statistics about the dataset

- File size

```
new_new-york_new-york.osm : 737,545KB
new_new-york_new-york.json: 832,593KB
```

- Number of tags

```
'node': 2961075
'relation': 2571
'way': 486020
```

Top 10 users who uploaded the data most often and frequency

```
import pprint

pipeline = [
    {'$group':{'_id': '$created.user',
'frequency': {'$sum':1} }}
    , {'$sort': {'frequency':-1}}
    , {'$limit':10}
    ]

result = db[collection].aggregate(pipeline)

for e in result:
    pprint.pprint(e)
```

```
{u'_id': u'Rub21_nycbuildings', u'frequency': 1630934}
{u'_id': u'ingalls_nycbuildings', u'frequency': 312022}
{u'_id': u'woodpeck_fixbot', u'frequency': 216254}
{u'_id': u'ediyes_nycbuildings', u'frequency': 90610}
{u'_id': u'lxbarth_nycbuildings', u'frequency': 78243}
{u'_id': u'NJDataUploads', u'frequency': 75882}
{u'_id': u'ingalls', u'frequency': 64129}
{u'_id': u'CoreyFarwell', u'frequency': 57218}
{u'_id': u'smlevine', u'frequency': 52717}
{u'_id': u'celosia_nycbuildings', u'frequency': 44876}
```

The number of data uploaded by year.

```
pipeline = [
            {'$project': {'year': { '$year': "$timestamp"}}}
             ,{'$group': {'_id':'$year', 'frequency': {'$sum': 1} }}
             ,{'$sort': {'_id':1}}
        ]

result = db[collection].aggregate(pipeline)

for r in result:
    pprint.pprint(r)
```

```
{u'_id': 2007, u'frequency': 1911}
{u'_id': 2008, u'frequency': 7496}
{u'_id': 2009, u'frequency': 341954}
{u'_id': 2010, u'frequency': 61920}
{u'_id': 2011, u'frequency': 33846}
{u'_id': 2012, u'frequency': 61844}
{u'_id': 2013, u'frequency': 1349442}
{u'_id': 2014, u'frequency': 1263142}
{u'_id': 2015, u'frequency': 203789}
{u'_id': 2016, u'frequency': 124226}
```

Finding the Five Nearest locations from 'AJ's Pizza'.

```
name = "AJ's Pizza"
pipeline ={'$query': {'name':name}
        ,'$project':{'name':'$name', 'pos':'$pos'}
       }
result = db[collection].find(pipeline)

for r in result:
    position = r['pos']


query = {'pos':{'$near':position}, 'name':{'$ne':None},
'tag_type':'node', 'pos':{'$ne':position}}

result = db[collection].find(query).limit(5)

for doc in result:
    print
    pprint.pprint(doc)
```

```
{u'id': u'3712775825',u'pos': [40.795324, -74.0471541]}
```

3. Conclusion

This project focused on improving data of major attributes, which are occupying more than 80% of data, and worked well enough to answer arisen questions in the project. However, there are still hundreds of field not cleaned yet. Especially, attributes that have sub categories such as gnis, tiger, cityrack seemed to have overlapping with other attributes. If we are given more time and enough information about these categories, we might try to improve the quality of data by cross-checking values of each attribute and integrating related attributes as many as possible.