OpenStreetMap Data Wrangling with MongoDB Project

Dmitry Tolstonogov

Problems Encountered in the Map

Map Area: New York, NY United States.
OSM XML file `new-york_new-york.osm` is obtained from MapZen . We created a smaller sample (1%) of the initial file `sample_1_pct 1.osm` for the exploratory analysis.

We would like to change the data model and expand the "addr:street" type of keys to a dictionary like this:
{"address": {"street": "Some value"}}
So, we have to see if we have such tags, and if we have any tags with problematic characters.

We modified the script `tags.py` from the problem Tag Types, Lesson 6 of Data Wrangling with MongoDB course in the following way:

> For the "k" value for each "<tag>" it checks if they can be valid keys in MongoDB, as well as check if there are any other potential problems.
> It uses regular expressions to look for certain patterns in the tags.
> The script returns the number of patterns and sets of samples for every pattern.

Using `explore_tags.py` we have found the following problems in the data:

1. There are multiple keys with the prefix like `"prefix:",` for example:
    a. `<tag k="addr:street" .../>`
    b. `<tag k="gnis:state_id" ... />`
    c. `<tag k="tiger:county" ... />`

    d. `<tag k="railway:atc" ... />`
    e. `<tag k="contact:fax" ... />`
    f. `<tag k="source:hgv" ... />`
    g. `<tag k="building:colour" ... />.`

2. There is cityrack information with dot separation, like
    a. `<tag k="cityracks.street" ... />;`
    b. We should change it similar to the before cases.
3. There are multiple references in `way` tags.
4. Some postal codes are incorrect; some are not belongs to New York City.
5. Some street names are inconsistent.

To deal with the problem 1, we changed the data model and expanded `"some_dict:some_key"` type of keys to a dictionary like this: `{"some_dict": {"some_key": "some_value"}}`. We did it for the all interesting cases discovered in our tag exploration.

Similarly, we did it for the problem 2 (cityrack).

To avoid multiple refs in "way" nodes like this:

```
  <nd ref="305896090"/>
  <nd ref="1719825889"/>
```

(problem 3) we turned into

```
"node_refs": ["305896090", "1719825889"]
```

The correct postcodes (problem 4) must have 5 digits and belong to the NYC area,
`10001 <= postcode <= 11692.`
Also, we corrected cases like `v = 'NY 10533' or v = '11229-8541'`

To deal with problem 5 (street name inconsistency), we modified the script from `audit.py` from the problem Improving Street Names, Lesson 6 of Data Wrangling with MongoDB course.

We join all the solutions above to `process_data.py` script. It parses `new-york_new-york.osm` file to `new-york_new-york.osm.json` file.

All the scripts passed the small sampled `sample_1_pct 1.osm` smoothly.
While parsing the big `new-york_new-york.osm` file, we have encountered some challenges.

First, while updating street names, we've got an exception in some place:
`UnicodeEncodeError: 'ascii' codec can't encode character u'\xf1' in position 8: ordinal not in range(128).` We resolved it applying the method `.encode('ascii', 'ignore').`

Second, we've got `MemoryError`. We got rid of it clearing each element after its shaping with `element.clear()` command.

Overview of the data

This section contains basic statistics about the dataset.

- **File sizes**

  `sample_1_pct 1.osm` is 21162 KB (about 300000 lines)

  `sample_1_pct 1.osm` is 23159 KB

  `new-york_new-york.osm` is 2094094 KB (about 30000000 lines).

  `new-york_new-york.osm.json` is 2315091 KB

- **Number of documents**

  ```
  db.count()

  Out[1]: 9831664
  ```

- **Number of unique users**

  ```
  len(db.distinct("created.user"))

  Out[2]: 2737
  ```

- **Number of nodes**

  ```
  db.find({"type":"node"}).count()

  Out[3]: 8439186
  ```

- **Number of ways**

```
db.find({"type":"way"}).count()
Out[4]: 1392478
```

- **Number of amenities**

```
db.find({"amenity": {"$exists" : 1}}).count()
Out[5]: 30505
```

- **Number of bridges**

```
len(db.distinct("bridge"))
Out[6]: 10
```

Indeed, NYC has 10 bridges.

Other ideas about the datasets

- **10 Top amenities**

```
pipeline = [{"$match" : {"amenity" : {"$exists" : 1}}},
{"$group" : {"_id" : "$amenity", "count": {"$sum" :1}}},
{"$sort" : {"count" : -1}}, {"$limit" : 10}]
res = db.aggregate(pipeline)
for item in res:
    pprint.pprint(item)
{u'_id': u'parking', u'count': 5135}
{u'_id': u'bicycle_parking', u'count': 4820}
{u'_id': u'school', u'count': 4628}
{u'_id': u'place_of_worship', u'count': 4491}
{u'_id': u'restaurant', u'count': 2280}
{u'_id': u'fast_food', u'count': 736}
{u'_id': u'cafe', u'count': 684}
{u'_id': u'fire_station', u'count': 667}
{u'_id': u'bank', u'count': 628}
{u'_id': u'bench', u'count': 477}
```

- **Starbucks cafes distribution by borough**

```
pipeline = [{"$match" : {"name" : {"$regex" : "Starbucks"}} },
\
{"$group" : {"_id" : "$address.city", "count": {"$sum" :1}}}, \
{"$sort" : {"count" : -1}}]
res = db.aggregate(pipeline)
for item in res:
    pprint.pprint(item)
{u'_id': None, u'count': 108}
{u'_id': u'New York', u'count': 38}
{u'_id': u'Brooklyn', u'count': 4}
{u'_id': u'Oakland', u'count': 1}
{u'_id': u'brooklyn', u'count': 1}
{u'_id': u'New York City', u'count': 1}
{u'_id': u'Flushing', u'count': 1}
{u'_id': u'Forest Hills', u'count': 1}
{u'_id': u'Ridgewood', u'count': 1}
{u'_id': u'Astoria', u'count': 1}
{u'_id': u'Cos Cob', u'count': 1}
{u'_id': u'Denville', u'count': 1}
```

Obviously, the number of Starbucks cafes in NYC is much larger. The result also outlines the inconsistency of addresses and a room for the further wrangling and improving of OpenStreetMap data.