OpenStreetMap Project

Data Wrangling with MangoDB

Lin LI

Map Area: Richmond Hill, York Region, Ontario, Canada

Table of Contents

1. Problems Encountered in the Map

After downloading the OSM XML file from Mapzen and running it against a provisional audit.py file, following porblems are encountered:

- Street name inconsistency
    - Direction names at the end
    - Abbreviated street types (eg. ‘Crt.’, ‘Rd’, ‘Dr’)
    - Missing street types (eg. ‘Yonge’, ‘Atlas Peak’, ‘Maple’)
    - Existence of unknown elements (eg. ‘Ste 500, #A3’)
    - Particular street (eg. ‘Highway 7’)
- Postal code
- Tags

Street Name Inconsistency

By running **audit.py**, I get a list of uncommon street types as you can see below.

Code for this part can be found in audit.py, functions audit_street_type, is_street_name, and audit. You can uncomment corresponding parts in test() to see the outputs.

Output:

```
{'500': set(['Allstate Parkway, Ste 500']),
 '7': set(['Highway 7']),
 'A3': set(['Rutherford Rd #A3']),
 'Close': set(['Campus Close']),
 'East': set(['7 Highway East',
              ...
              'Weldrick Road East']),
 'Maple': set(['Maple']),
 'North': set(['Bridgeford Street North',
              ...
             'Taylor Mills Drive North']),
 'Peak': set(['Atlas Peak']),
 'Point': set(['Fox Point']),
 'South': set(['Bridgeford Street South',
              ...
              'Taylor Mills Drive South']),
 'West': set(['Bloomington Rd West',
              ...
             'Weldrick Road West']),
 'Yonge': set(['Yonge'])}
```

In Richmond Hill, it’s normal to have **direction names** (East, West, North, South) at the end of a street, so I will keep it at the end.

However, I have noticed that there are **over abbreviated** street names or directions names in between. For example:

```
'W': set(['Bloomington Rd W'])
```

Some streets doesn’t have street types written. For example:

```
'Yonge': set(['Yonge']),
'Peak': set(['Atlas Peak']),
'Maple': set(['Maple'])
```

Following changes are made to these streets:

```
Yonge => Yonge Street
Maple => Maple Street
Atlas Peak => Atlas Peak Drive
```

A particular case is ‘Highway 7’, it’s written as ‘Highway 7’ or ‘7 Highway’ in OSM XML. By convention it’s referred as ‘Highway 7’ in Richmond Hill, however, for the purpose of better querying experience in the future I decided to keep it as ‘7 Highway’.

```
Highway 7 => 7 Highway
```

There are unknown elements attached in street names, examples:

```
'500': set(['Allstate Parkway, Ste 500']),
 'A3': set(['Rutherford Rd #A3'])
```

I decided to simply remove them:

```
Allstate Parkway, Ste 500 => Allstate Parkway
Rutherford Rd #A3 => Rutherford Road
```

One street that I don’t recognize and couldn’t find after some research is ‘Fox Point’, so I decided to put the FIXME which follows the convention as discovered in this dataset.

```
Fox Point => FIXME
```

Code of this part is in audit.py, function update_name(name, mapping).

Post Code

In Richmond Hill, postal code format is “LXX XXX” with a space inbetween. In the dataset, some of the post code doesn’t have space in between. And 3 special cases (L4K, M6A2T9, L4B) are ignored due to incompleteness and M6A2T9 is postal code from Toronto.

Tags

After running **tags.py**, we can get the output:

```
===== Tag Types Count =====

{'FIXME': 13,
 'lower': 142609,
 'lower_colon': 90440,
 'lower_digit': 11,
 'other': 3364,
 'problemchars': 0,
 'lower_three_colons': 8,
 'lower_two_colons': 144}
```

We can see that the tags are really messy

Problems and Solutions

“FIXME” will be ignored

“lower_dig” (eg. ‘ref_1’, ‘building_1’, ‘source_1’, ‘phone_1’) tags have the base names with some lower tags (eg. ‘ref’, ‘building’, ‘source’), however with numbers attached at the end. Need to change the name to lower tags names and create a list for the values and add the value to the list. Eg:

```
set(['ref_1', 'phone_1', 'building_1', 'source_1'])
```

“lower_three_colons”. Eg:

```
set(['parking:lane:both:parallel'])
```

“lower_two_colons”. Eg:

```
set(['parking:lane:right', 'destination:ref:to', 'parking:condition:both',
'parking:lane:both', 'turn:lanes:forward', 'turn:lanes:backward',
'destination:street:to'])
```

For the purpose of this project, “lower_three_colons” and “lower_two_colons” will be ignored.

“lower_colon” (A:B): there are tags where A can be found in single “lower” tags. For example (“name” and “name:fr”), these tags will also be ignored. Following is the list of these tags.

```
duplist = ["colour", "name", "atm", "building", "source", "destination", "lanes",
"railway","maxspeed","opening_hours","hov","internet_access","phone",
"aerialway","capacity"]
```

“Other” tags: “fuel:octane_98”, “fuel:octane_95”, and “fuel:HGV_diesel” will be ignored because there are similar fields in “lower_colon” tags already. “geobase:...” will be saved in the dictionary. Other fields will be ignored.

```
defaultdict(<type 'int'>, {'canvec:CODE': 185, 'geobase:acquisitionTechnique': 2735,
'fuel:HGV_diesel': 1, 'canvec:UUID': 203, 'addr:city_1': 1, 'canvec:CODE:1580010': 23,
'is_in:iso_3166_2': 1, 'fuel:octane_98': 1, 'geobase:datasetName': 213,
'fuel:octane_95': 1})

==========ignored tags===========

set(['canvec:CODE', 'fuel:HGV_diesel', 'canvec:UUID', 'destination:ref:to',
'canvec:CODE:1580010', 'parking:condition:both', 'is_in:iso_3166_2', 'fuel:octane_98',
'parking:lane:both:parallel', 'parking:lane:both', 'turn:lanes:forward',
'parking:lane:right', 'turn:lanes:backward', 'destination:street:to', 'fuel:octane_95',
'FIXME'])
```

2. Data Overview

File sizes

```
richmondhill.osm .......... 58.7MB
richmondhill.osm.json ..... 65.2MB
```

Basic Statistics

```
# Number of documents
> db.osm.find().count()
281755
# Number of nodes
> db.osm.find({"type":"node"}).count()
246404
# Number of ways
> db.osm.find({"type":"way"}).count()
35251
```

Contributor Statistics

```
# Number of unique users
> db.osm.distinct("created.user").length
227
# Top 10 contributing user
> db.osm.aggregate([{$group:{_id:"$created.user", count: {$sum: 1}}},{$sort:{count:
-1}},{$limit:10}])
{ "_id" : "andrewpmk", "count" : 233840 }
{ "_id" : "geobase_stevens", "count" : 8333 }
{ "_id" : "Mojgan Jadidi", "count" : 6730 }
{ "_id" : "mighty_snake", "count" : 3832 }
{ "_id" : "petrar_telenav", "count" : 3707 }
{ "_id" : "istvanv_telenav", "count" : 3639 }
{ "_id" : "WhiskyMacK", "count" : 3281 }
{ "_id" : "bdiscoe", "count" : 2342 }
{ "_id" : "Kevo", "count" : 1238 }
{ "_id" : "Conan Chan", "count" : 1154 }

# Number of users apprearing only once (having 1 post)
> db.osm.aggregate([{$group:{_id:"$created.user", count:{$sum: 1}}}, {$group:{_id:"$count",
num_users: {$sum:1}}}, {$sort:{_id:1}}, {$limit:1}])
{ "_id" : 1, "num_users" : 57 }
# "_id" represents postcount
```

The contribution of users seems incredibly skewed. Possibly due to automated versus manual map editing. Here are some user percentage statistics:

- Top user contribution (“andrewpmk”) : 82.99%
- Combined Top 10 users contribution : 95.15%
- Number of users making up the rest 4.85% of contribution : 217 users
- 25.1% users appeared only once

The fact that most users contribute very little may due to lack of motivating force. Gamification could be applied to encourage engagement of contributors. For example, in applications like Google Maps/ Yelp users gain reputation (gamification elements: rewards, badges, or a leaderboard) from contributing by writing reviews, tagging locations, etc. Top contributors would also be invited to community events.

3. Additional Ideas

Additional data exploration using MongoDB queries

```
# Top 10 appearing amenities
> db.osm.aggregate([{$match:{amenity:{$exists:1}}}, {$group:{_id:"$amenity",count:{$sum:1}}},
{$sort:{count:-1}},{$limit:10}])
{ "_id" : "parking", "count" : 1345 }
{ "_id" : "post_box", "count" : 120 }
{ "_id" : "school", "count" : 103 }
{ "_id" : "restaurant", "count" : 98 }
{ "_id" : "fast_food", "count" : 73 }
{ "_id" : "bank", "count" : 55 }
{ "_id" : "bench", "count" : 51 }
{ "_id" : "cafe", "count" : 49 }
{ "_id" : "fuel", "count" : 43 }
{ "_id" : "place_of_worship", "count" : 43 }

# Parking lot types
> db.osm.aggregate([{$match:{amenity:{$exists:1}, amenity: "parking"}},
{$group:{_id:"$parking", count: {$sum:1}}}, {$sort:{count:-1}}, {$limit:10}])
{ "_id" : "surface", "count" : 1289 }
{ "_id" : "underground", "count" : 25 }
{ "_id" : null, "count" : 23 }

# Top cuisine in Richmond Hill
> db.osm.aggregate([{$match:{amenity:{$exists:1}, amenity: "restaurant"}},
{$group:{_id:"$cuisine", count: {$sum:1}}}, {$sort:{count:-1}}, {$limit:10}])
{ "_id" : null, "count" : 54 }
{ "_id" : "pizza", "count" : 5 }
{ "_id" : "american", "count" : 5 }
{ "_id" : "chinese", "count" : 4 }
{ "_id" : "asian", "count" : 3 }
{ "_id" : "indian", "count" : 3 }
{ "_id" : "sushi", "count" : 3 }
{ "_id" : "japanese", "count" : 3 }
{ "_id" : "thai", "count" : 2 }
{ "_id" : "chicken", "count" : 2 }
{ "_id" : "multi-storey", "count" : 8 }

# Top Fast Food Restaurants in Richmond Hill
> db.osm.aggregate([{$match:{amenity:{$exists:1}, amenity: "fast_food"}},
{$group:{_id:"$name", count: {$sum:1}}}, {$sort:{count:-1}}, {$limit:10}])
{ "_id" : "Subway", "count" : 8 }
{ "_id" : "McDonald's", "count" : 8 }
{ "_id" : "Harvey's", "count" : 4 }
{ "_id" : "Mr. Sub", "count" : 4 }
{ "_id" : "Pizza Nova", "count" : 4 }
{ "_id" : "Hero Certified Burgers", "count" : 3 }
{ "_id" : "Pizza Pizza", "count" : 3 }
{ "_id" : "Burger King", "count" : 2 }
{ "_id" : "DQ Restaurant", "count" : 2 }
{ "_id" : "Wendy's", "count" : 2 }

# Top Cafes in Richmond Hill
> db.osm.aggregate([{$match:{amenity:{$exists:1}, amenity: "cafe"}}, {$group:{_id:"$name",
count: {$sum:1}}}, {$sort:{count:-1}}, {$limit:10}])
{ "_id" : "Tim Hortons", "count" : 21 }
{ "_id" : "Coffee Time", "count" : 4 }
{ "_id" : "Starbucks Coffee", "count" : 4 }
{ "_id" : "Second Cup", "count" : 3 }
{ "_id" : "Country Style", "count" : 2 }
{ "_id" : "Destiny", "count" : 1 }
{ "_id" : "TenRen's Tea Time", "count" : 1 }
{ "_id" : "Neighbours Coffee", "count" : 1 }
{ "_id" : "Yofee Coffee", "count" : 1 }
{ "_id" : null, "count" : 1 }


# Top religion in Richmond Hill
 > db.osm.aggregate([{$match:{amenity:{$exists:1}, amenity: "place_of_worship"}},
{$group:{_id:"$religion", count: {$sum:1}}}, {$sort:{count:-1}}, {$limit:3}])
{ "_id" : "christian", "count" : 36 }
{ "_id" : "jewish", "count" : 3 }
{ "_id" : null, "count" : 3 }
```

From the above queries, we can see that the most common type of amenities in Richmond Hill is parking, and mostly surface parking. It makes sense because Richmond Hill is a suburban area with mostly residences. We have 21 Tim Hortons, which shouldn't be too surprising. Due to the large Asian community in this area, we can also see Asian restaurants and tea-shops in the list. Top religion in Richmond Hill is Christian.

Conclusion

When I first got the dataset, it was not really dirty. Only few changes needed to be made on street names and postal codes. What got me struggling was the problem of the tags, there are a lot of tags that seem really unclear. After getting a bit familiar with the tags, I decided to ignore some unclear tags for the purpose of the project. However, some of them do seem like useful data, I would love to know more about these tags and build a better structure of data.
After reviews of the data using MongoDB, it's obvious that Richmond Hill area is incomplete. Even though it's not the busiest city, but we do have amenities more than recorded in OSM. For example, we have a large community of Hindus which is not recorded in the dataset.

Improvements can be made on datasets:

- Gamification
- Connect with the popular game Pokemon Go, or Yelp

Benefits:

- Motivates user contribution
- Pokemon Go users provides accurate GPS location of interesting points (Pokestops, etc.)
- Large user base can generate a lot of data really quickly
- Yelp has large database on interesting locations (restaurants, shops, etc.) and their specific information (cuisine, price, etc.)

Anticipated Problems:

- Pokestops don't really represent interesting locations in real life
- Platforms like Yelp or Pokemon Go may be protective of their data
- Data format / structure may be different between these platforms, need to reformat the structures