OpenStreetMap Data Wrangling Case Study – Jackie Tang

Map Area
Toronto, ON, Canada

- https://www.openstreetmap.org/relation/324211

The map above is of my hometown, though the area that I selected from mapzen.com represents the Greater Toronto Area which includes neighbouring municipalities to those in the above map boundary. I am interested in what database querying reveals about the data set, and hopefully some of my analyses will help me to gain a much better understanding of my own city!

Problems Encountered in the Map

After initially downloading a small sample size of the full map of the Greater Toronto Area, I noticed four main problems with the data, which I will discuss in the following order.

- Lower-case letters in postal codes (“M2n 6k7”)
- Inconsistent spacing within postal codes (“M2N6K7”)
- Unnecessary address fields (“addr: interpolation”)
- Regional coverage issues (Places such as Youngstown, NY and Hamilton, ON are included)

Auditing and cleaning the Toronto OSM data set

Cleaning postal codes:

While doing some basic querying on the sample file, I noticed first hand that there were issues with the postal codes wherein some letters were lower case. I wrote two auditing/cleaning functions named clean_postcode and check_lower_letters, which first converts every postcode into upper letters and then does a separate check to see if there are any postcodes left with lower case letters. I also re-use the code in clean_postcode later in the shape_element function for outputting to a JSON file. To make things simple, the code for this is shown in the script open_street_map.py,

However, after insertion into the MongoDB database, a query for the top 10 postcodes revealed that there were postcodes in which there was no space between the first 3 and last 3 characters of the postal code (it is customary in Canada to have postcodes in the form of A9A 9A9).

```
If key == “postcode”:
    if address[key][3:4] != " ":
        address[key] = address[key][:3] + " " + address[key][3:]
```

The piece of code above was added into the shape_element (cleaning function) to add a space after the third character in a postcode if the 4th character was not a space.

Unnecessary address fields:
While still working within the subset file, I noticed that the top 2 addresses had only the fields "interpolation" = "odd" and "interpolation" = "even" as an address field, with no other fields such as city, house number, street address, postcode in the address. As they would always show up as the top 2 addresses and were irrelevant to my analysis, these fields were removed using the short piece of code below from all nodes in the JSON file.

```
for subtag in element:
        if subtag.tag == 'tag':
        if subtag.attrib['k'] == 'addr:interpolation':
                continue
```

Regional coverage issues:
After insertion into the database, the address list showed many cities/towns incorporated within the Greater Toronto Area (GTA) that were not supposed to be added into the OSM data. A list of these cities was compiled and later verified with the standard definitions of what is included in the Greater Toronto Area and what was not. An exhaustive search was done to ensure that close to all cities in the data were to be a part of the GTA, and those that did not fit the definition were put in a list named city_list, and those that were not part of that list had the entire node removed from outputting into the JSON file.

Following from the previous piece of code above:

```
elif subtag.attrib['k'] == 'addr:city':
        if subtag.attrib['v'] in city_list:
        return
```

Summary statistics:

Compressed XML file size: 83 MB
Uncompressed file size: 1.2 GB

Number of nodes: 5,856,855 (before cleaning), 5,781,333 (after cleaning)

```
documents_num = coll.find().count()
```

Number of ways: 5,098,141 (before), 5,023,138 (after)

```
nodes_num = coll.find({"type": "node"}).count()
```

Number of unique contributors: 2743 (before), 2698 (after)

```
users = len(coll.distinct("created.user"))
```

Interesting statistics on user contribution:

Top contributor : andrewpmk – 57.6% of total entries
Top 10 contributors: 84.1% of total entries

{u'count': 3330783, u'_id': u'andrewpmk'}
{u'count': 484121, u'_id': u'Kevo'}
{u'count': 470568, u'_id': u'MikeyCarter'}
{u'count': 207884, u'_id': u'Bootprint'}
{u'count': 142820, u'_id': u'Victor Bielawski'}
{u'count': 97272, u'_id': u'Mojgan Jadidi'}
{u'count': 77867, u'_id': u'geobase_stevens'}
{u'count': 75127, u'_id': u'rw__'}
{u'count': 38491, u'_id': u'Gerit Wagner'}
{u'count': 34237, u'_id': u'brandoncote'}

```
cursor = coll.aggregate([{"$match":{"created.user":{"$exists":1}}},
            {"$group":{"_id":"$created.user", "count":{"$sum":1}}},
            {"$sort":{ "count" : -1}},  {"$limit":10}])
```

Top ten appearing amenities:

{u'count': 35461, u'_id': u'parking'}
{u'count': 3426, u'_id': u'fast_food'}
{u'count': 3299, u'_id': u'restaurant'}
{u'count': 2574, u'_id': u'school'}
{u'count': 2435, u'_id': u'bench'}
{u'count': 2041, u'_id': u'post_box'}
{u'count': 1940, u'_id': u'place_of_worship'}
{u'count': 1694, u'_id': u'cafe'}
{u'count': 1289, u'_id': u'bank'}
{u'count': 1201, u'_id': u'waste_basket'}

```
cursor = coll.aggregate([{"$match":{"amenity":{"$exists":1}}},
            {"$group":{"_id":"$amenity", "count":{"$sum":1}}},
            {"$sort":{ "count" : -1}},  {"$limit":10}])
```

It is very surprising that there are only 1201 waste baskets and 2435 benches, perhaps users are not interested in mapping for less important amenities like these ones!

Top 10 Shops:

{u'count': 1503, u'_id': u'convenience'}
{u'count': 1195, u'_id': u'clothes'}
{u'count': 826, u'_id': u'hairdresser'}
{u'count': 817, u'_id': u'supermarket'}
{u'count': 703, u'_id': u'vacant'}
{u'count': 646, u'_id': u'beauty'}
{u'count': 583, u'_id': u'car_repair'}
{u'count': 538, u'_id': u'car'}
{u'count': 400, u'_id': u'dry_cleaning'}
{u'count': 351, u'_id': u'alcohol'}

```
cursor = coll.aggregate([{"$match":{"shop":{"$exists":1}}},
            {"$group":{"_id":"$shop", "count":{"$sum":1}}},
            {"$sort":{ "count" : -1}}, {"$limit":10}])
```

It seems that there are quite a few vacant shops in Toronto, which might be due to businesses having gone out of business in the OSM data, and instead a new entry was created for a new business replacing the former location of that business. Another reason might be that the entry did not have an address.

Top 10 Cuisines:
{u'count': 993, u'_id': u'coffee_shop'}
{u'count': 636, u'_id': u'pizza'}
{u'count': 533, u'_id': u'sandwich'}
{u'count': 508, u'_id': u'burger'}
{u'count': 234, u'_id': u'chinese'}
{u'count': 170, u'_id': u'chicken'}
{u'count': 135, u'_id': u'indian'}
{u'count': 128, u'_id': u'japanese'}
{u'count': 112, u'_id': u'italian'}
{u'count': 95, u'_id': u'ice_cream'}

```
cursor = coll.aggregate([{"$match":{"cuisine":{"$exists":1}}},
            {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
            {"$sort":{ "count" : -1}}, {"$limit":10}])
```

Top 10 Buildings:
{u'count': 61160, u'_id': u'yes'}
{u'count': 15088, u'_id': u'house'}
{u'count': 6805, u'_id': u'retail'}

{u'count': 6627, u'_id': u'residential'}
{u'count': 5648, u'_id': u'apartments'}
{u'count': 4356, u'_id': u'industrial'}
{u'count': 1761, u'_id': u'school'}
{u'count': 1135, u'_id': u'church'}
{u'count': 991, u'_id': u'office'}
{u'count': 867, u'_id': u'garage'}

```
cursor = coll.aggregate([{"$match":{"building":{"$exists":1}}},
        {"$group":{"_id":"$building", "count":{"$sum":1}}},
        {"$sort":{ "count" : -1}}, {"$limit":10}])
```

Buildings – Investigative analysis

From looking at data on the top 10 buildings, it can be seen that the top building by far is “yes” at over 60,000 entries, which could represent any type of building. Unsure of what “yes” represented, I wrote the following query below to find the top 5 buildings in which were “yes” types.

```
bldg = coll.find({"building": "yes"}).limit(5)
```

Results:
{u'building': u'yes', u'name': u'Nona MacDonald Visitors Centre'}
{u'building': u'yes', u'name': u'Fedex Warehouse'
{u'building': u'yes', u'name': u'Picaddily Outlet'
{u'building': u'yes', u'name': u'Student Centre', u'note': u'Not sure if sales outlet or office building'
{u'building': u'yes', u'note': u'utility building'

Looking at the names and notes of the first few entries, we see that a lot of these buildings may in fact be community centres, warehouses, shelters or utilities that should have been classified under those particular categories. Clearly, a way to improve the OpenStreeMap data for Toronto would be to write statements such that if “name” contain the words: arena, warehouse, community centre, that these be categorized into their own respective categories under either the building and/or amenity/shop fields. One anticipated problem would that it would require a huge manual inspection effort in order to come up with the appropriate categories to output to. However, the benefits would be enormous as we would get a much better picture of the variety and number of amenities, shops and buildings in the GTA.

I looked at the top 10 buildings, and I noticed that there was also a serious undercount of single-detached homes, as data shows there were only around 15,000 houses in the entire data set covering the Toronto metro area. This is not something that can be cleaned programmatically, but having contributors making better efforts to map residential homes would be a good suggestion.

Top 10 Cities:

{u'count': 111388, u'_id': u'City of Toronto'}
{u'count': 37166, u'_id': u'Mississauga'}
{u'count': 25393, u'_id': u'City of Brampton'}
{u'count': 17311, u'_id': u'City of Vaughan'}
{u'count': 16798, u'_id': u'Town of Markham'}
{u'count': 15784, u'_id': u'Oakville'}
{u'count': 11468, u'_id': u'Richmond Hill'}
{u'count': 10978, u'_id': u'City of Burlington'}
{u'count': 10644, u'_id': u'Town of Whitby'}
{u'count': 9128, u'_id': u'Toronto'}

```
cursor = coll.aggregate([{"$match":{"address.city":{"$exists":1}}},
            {"$group":{"_id":"$address.city", "count":{"$sum":1}}},
            {"$sort":{ "count" : -1}}, {"$limit":100}])
```

Address field: Cities – Investigative analysis

Even after correcting for the regional coverage issues earlier, I later saw there were separate/duplicate entries for “City of Toronto” and “Toronto”, or “Town of Markham” versus “Markham”. Further, former municipalities that have been incorporated into Toronto such as North York, Scarborough, Etobicoke, and East York are seen as separate cities. There were a few instances of neighbourhoods within the city of Toronto such as “Willowdale” which is also within the former town of “North York” or “Oak Ridges” which is a community within Richmond Hill (also part of the GTA).

To avoid non-duplication of address fields, and to make for more effective querying, I would suggest cleaning the code so that instances of these communities/neighbourhoods are labeled according to their town/city jurisdiction, and that all instances of the former towns/cities which were amalgamated into the GTA or its duplicate name “Toronto” be renamed to fall under the “City of Toronto” (which is in by far the largest number of nodes). When OSM users want to query for the number of amenities or shops within a particular sub-region of the GTA, users would not have to account for every possible duplicate name or community within that jurisdiction. That being said, the one major problem I can see from this is that it would be difficult to always keep up with all the misspellings and random names that users input into a city field within the address (ex. Torxonto, Taranto etc.).

Conclusion

After this review of the data, it’s obvious that the Toronto area is incomplete, though I believe it has been well cleaned for the purposes of this exercise. Given the size of Toronto, it might take a while to really get the data quality improved, but I believe it could be possible!