Udacity NanoDegree: Data Analysis
Project 2: Data Wrangling With MongoDB

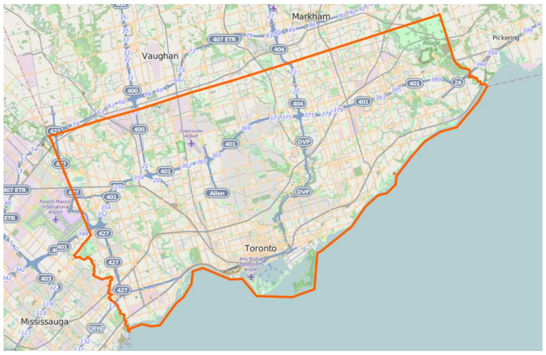
Submitted by: Rakesh Dhote
Email: rakesh.dhote@gmail.com

Contents

Problem Statement:

Use data munging techniques, such as assessing the quality of the data for validity, accuracy, completeness, consistency and uniformity, to clean a map area in the OpenStreetMap (OSM) data.

Map Area: Toronto, Ontario, Canada
Location: latitude(43.201, 44.182), longitude(-78.717, -80.16)



Brief overview of the project:

The Toronto map extract downloaded from the Map Zen. The Toronto city is selected for this project as it is a financial hub of Canada and has the biggest size Canadian city map extract available on the Map Zen. In the following sections, details about the problems encountered, map details, and additional ideas about the data set are presented.

Problems encountered during the project and their resolutions:

The Toronto map extract is relatively clean. Following are the main challenges encountered during the data auditing process and the actions taken to resolve them:

Inconsistencies in postcode:

A Canadian postcode is a six-character alphanumeric string in the form *A1A 1A1*, where *A* is an upper letter, *1* is a digit, and a space separating the third and fourth characters [1]. A postcode is typically unique to a very small area, such as 1-5 private homes, a single apartment building or business, etc. [2].

On querying the Toronto map data, the following inconsistencies are observed in the postcodes:

- All small letter characters (ex. m4y 1r5),
- No space separating the third and fourth characters (ex. M4Y1R5),
- Multiple postcode for the same location (ex. M5T 1R9, M1P 2L7),

The Regex expressions are used to clean the postcode to the correct format (ex. M4Y 1R5) using the Python script before dumping the data to a JSON file.

Inconsistencies in city name:

As like other countries, cities in Canada are also known by their alternative name(s). For example, the Toronto is as also well known as the *City of Toronto*. While auditing the data, it is recommended to use the same city name. The following key:value (old:new) city pairs are used to update the city name. Total 64 documents with the city field entries as '?' are deleted from the collection.

'Ajax, Ontario' : 'Ajax',	'Town of East Gwillimbury' : 'East Gwillimbury',
'caledon' : 'Caledon',	'Town of Erin' : 'Erin',

```
'City of Brampton' : 'Brampton',
'City of Burlington' : 'Burlington',
'City of Hamilton' : 'Hamilton',
'City of Kawartha Lakes' : 'Kawartha Lakes',
'City of Oshawa' : 'Oshawa',
'City of Pickering' : 'Pickering',
'City of St. Catharines' : 'St. Catharines',
'City of Toronto' : 'Toronto',
'City of Vaughan' : 'Vaughan',
'Etobicoke, Toronto' : 'Etobicoke',
'Missisauga' : 'Mississauga',
'King' : 'King City',
'Municipality of Clarington' : 'Clarington',
'markham' : 'Markham',
'toronto' : 'Toronto',
'vaughan' : 'Vaughan',
'Town of Ajax' : 'Ajax',
'Town of Aurora' : 'Aurora',
'Town of Bradford West Gwillimbury' : 'Bradford West Gwillimbury',
'Town of Caledon' : 'Caledon',
'Town of Grimsby' : 'Grimsby',
'Town of Halton Hills' : 'Halton Hills',
'Town of Innisfil' : 'Innisfil',
'Town of Markham' : 'Markham',
'Town of Milton' : 'Milton',
'Town of Mono' : 'Mono',
'Town of New Tecumseth' : 'New Tecumseth',
'Town of Newmarket' : 'Newmarket',
'Town of Niagara-On-The-Lake' : 'Niagara-on-the-lake',
'Town of Whitby' : 'Whitby',
'Town of Whitchurch-Stouffville' : 'Whitchurch-Stouffville',
'Township of Adjala-Tosorontio' : 'djala-Tosorontio',
'Township of Amaranth' : 'Amaranth',
'Township of East Garafraxa' : 'East Garafraxa',
'Township of Essa' : 'Essa',
'Township of Guelph/Eramosa' : 'Guelph/Eramosa',
'Township of King' : 'King City',
'Township of Mulmur' : 'Mulmur',
'Township of Puslinch' : 'Puslinch',
'Township of Scugog' : 'Scugog',
'Township of Uxbridge' : 'Uxbridge'
```

Inconsistencies in street name:

The map extract for Toronto is relatively clean. Among ~6 million data points (nodes, ways, relations), approximately 100 street names needed cleaning. The data cleaning is achieved via a Python script using the Regex expressions. Following are the key:value (old:new) street pairs are used to update the tail part of the street name.

```
'Ave': 'Avenue',
'Ave.': 'Avenue',
'avenue' : 'Avenue',
'Alliston': '',
'Amaranth': '',
'Avens': 'Avens Boulevard',
'Blvd': 'Boulevard',
'Blvd.': 'Boulevard',
'Boulevade': 'Boulevard',
'Cir': 'Circle',
'Crct': 'Crescent',
'Cresent': 'Crescent',
'Cressent': 'Crescent',
'Crt.': 'Circuit',
'Dr': 'Drive',
'Dr.': 'Drive',
'E': 'East',
'E.': 'East'
'Grv': 'Grove',
'Ldg': 'Landing',
'Hrbr': 'Harbour Way',
'Manors': 'Manor',
'N': 'North',
'Puschlinch': 'Puslinch',
'Rd': 'Road',
'Rd.': 'Road',
'S': 'South',
'S.': 'South',
'St': 'Street',
'St.': 'Street',
'Terace': 'Terrace',
'Terraces': 'Terrace',
'Trl': 'Trail',
'W': 'West',
'W.': 'West'
```

Map extract covers a greater area than the Toronto city:

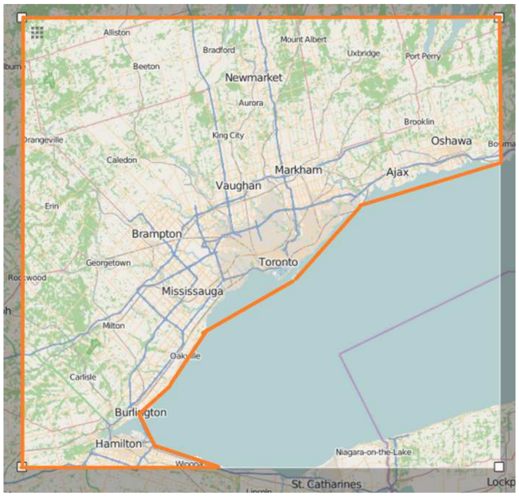
Though the map extract is downloaded for the Toronto city, initial queries using the bash *less* and *grep* commands indicated that the map region is not limited to the Toronto (or Greater Toronto Area). A MongoDB query

```
db.toronto.distinct('address.city')
```

suggests that the map region encompasses neighboring 107 suburbs as indicated in the right figure.

Keswick
Alliston
Bradford
Mount Albert
Uxbridge
Port Perry
Shelburne
Beeton
Newmarket
Aurora
Brooklin
Oshawa
Bowmanville
King City
Orangeville
Caledon
Markham
Ajax
Vaughan
Erin
Brampton
Toronto
Georgetown
Rockwood
Mississauga
Milton
Oakville
Carlisle
Burlington
Hamilton
Winona
Niagara-on-the-Lake
St Catharines
Lockport

[Left] Assumed Toronto map extract



[Right] Actual Toronto map extract along with the neighboring suburbs

Overview of the data:

File Sizes:

toronto_canada.osm ----------------- 1.14 GB
toronto_canada.osm.json ------------ 1.22 GB

Document Statistics:

# of documents:

```
db.toronto.count()
6006836
```

# of nodes

```
db.toronto.find({type:'node'}).count()
5370793
```

# of ways

```
db.toronto.find({type:'way'}).count()
636043
```

Regional Statistics:

# of cities in the collection

```
db.toronto.distinct('address.city').length
107
```

# of items (nodes/ways) in the top five cities

```
db.toronto.aggregate([ {   $match:{'address.city': {'$exists':1} } }, { group:{ '_id' : '$address.city', ‘count' : {$sum: 1 } } }, { $sort : { 'count' : -1 } }, { $limit : 5 } ])
```

_id	count
Toronto	120590
Hamilton	40553
Mississauga	36746
Brampton	27704
Markham	18606

# top three nodes sharing the same postcode

```
db.toronto.aggregate([ { $match:{'address.postcode': {'$exists':1} } }, { $group:{ '_id' :
'$address.postcode', 'count' : {$sum: 1 } } }, { $sort : { 'count' : -1 } }, { $limit : 3 } ])
```

_id	count
L7M 4A9	54
M6P 2W9	43
L6S 4B1	42

A quick Google search indicted that all the top three nodes are the residential areas.

# average nodes that share same postcode greater than 5 times

```
db.toronto.aggregate([{$match:{'address.postcode': {'$exists':1} }, {$group:{ '_id' :
'$address.postcode', 'count' : {$sum: 1 } } }, {$match : {'count' : {$gt : 5}} }, {$group:{ '_id' : null,
'average' : {$avg: '$count' } } } ])
```

14.45 ~ 14 nodes

# of documents where the postcode starts with M or L

```
db.toronto.aggregate([ { $match:{'address.postcode': /^M/ } }, { $group:{ '_id' : null, 'count' :
{$sum: 1 } } } ])
```

M: 2638

L: 2375

# nodes that have a single unique postcode

```
db.toronto.aggregate([{$match:{'address.postcode': {'$exists':1} }, {$group:{ '_id' :
'$address.postcode', 'count' : {$sum: 1 } } }, {$match : {'count' : 1} }, {$group:{ '_id' : null,
'sumone' : {$sum: 1 } } } ])
```

1894

# top 3 common housenames

```
db.toronto.aggregate([ { $match:{'address.city': {'$exists':1} } }, { $group:{ '_id' :
'$address.housename', 'count' : {$sum: 1 } } }, { $sort : { 'count' : -1 } }, { $limit : 3 } ])
```

_id	count
Overlea Plaza	9
Centerpoint Mall	2
Sydenham Place	2

Contributor Statistics:

# of unique users

```
db.toronto.distinct('created.user').length
```

1548

# of documents by top 3 contributers

```
db.toronto.aggregate([ { $group:{'_id':'$created.user', 'count':{$sum:1}} }, { $sort : { 'count' :-1} },
{ $limit : 3 } ])
```

_id	count
andrewpmk	4075384
MikeyCarter	486932
Kevo	374845

From a quick glance at the entries from the top three users, it is observed that the document entries follow a similar pattern. Probably the contributors have used an automated way of auditing and uploading the OSM data.

# of users having one entry

```
db.toronto.aggregate([{$group:{ '_id' : '$created.user', 'count' : {$sum: 1 } } }, {$match : {'count' : 1} }, {$group:{ '_id' : null, 'sumone' : {$sum: 1 } } } ])
312
```

# of items (nodes/ways) revised more than 5 times

```
db.toronto.aggregate([{$group:{ '_id' : '$created.version', 'count' : {$sum: 1 } } }, {$match : {'count' : {$gt : 5}} }, {$group:{ '_id' : null, 'countd' : {$sum: '$count' } } } ])
6006776
```

# of items (nodes/ways) revised once (only added to the collection)

```
db.toronto.aggregate([{$group:{ '_id' : '$created.user', 'count' : {$sum: 1 } } }, {$match : {'count' : 1} }, {$group:{ '_id' : null, 'sumone' : {$sum: 1 } } } ])
5098647
```

Additional ideas about the data set

During the data analysis, documents with the *city* entries '?' are removed. The documents can be retained by updating the *city* entry using the MongoDB *$near* or *$geonear* command to match the closest *city* based on the longitude and latitude.

Though the Toronto map extract is relatively clean, a preliminary data analysis indicates that it still needs further enrichment through GIS and geo-tagging. Possible approaches to the information enrichments are:

- Using social media to tag node location to an appropriate landmark, building, eatery, etc.
- Using API's (such as Yelp, Yellow pages, etc.) to geo-tag the node/way/relation data.

Such enriched information can be rolled back to the OSM database. The database can be used to develop interesting mobile and web applications such as tracking real-time noise, air, and sound pollution using the OSM API.

Conclusions

In summary, there are more than 6 million data entries in the Toronto OSM extract. The data is relatively clean with a small fraction of inconsistent data entries in postcode, street names, and cities. The data is audited with the Python script followed by exporting the XML data to the JSON for ingestion in the MongoDB. The document statistics reveal that further enrichment of data is required for developing interesting web and mobile applications using the OSM API.

References:

[1] https://en.wikipedia.org/wiki/Postal_codes_in_Canada
[2] https://ca.answers.yahoo.com/question/index?qid=20110709221829AA83Lks