Udacity Data Analyst NanoDegree Project

Data Wrangling: Wrangle OpenStreetMap Data

Student: Rob Dods

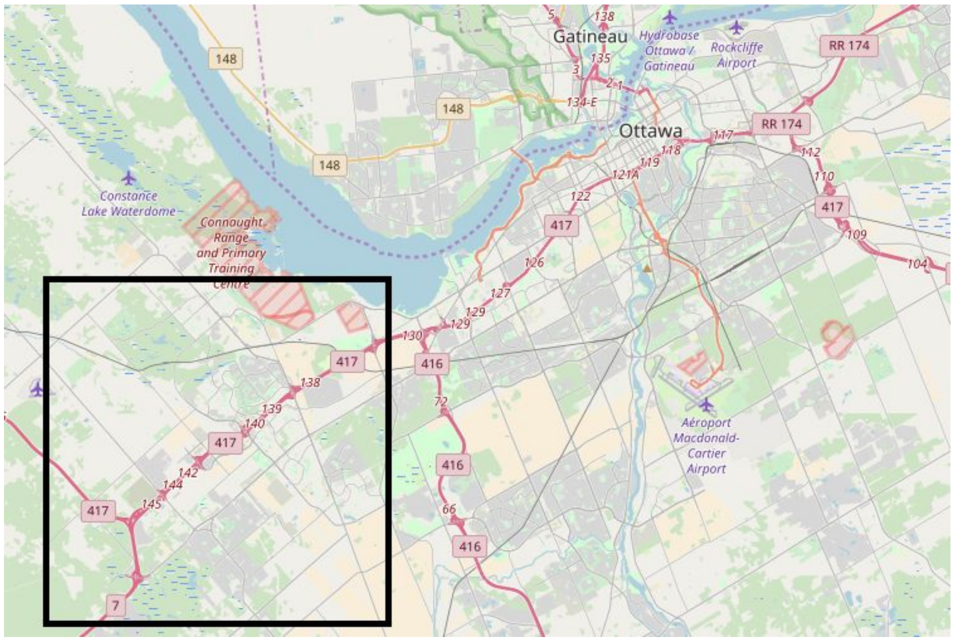
Map Area: Kanata, Ontario, Canada

A - <u>Project Goal</u>

To choose any area of the world in https://www.openstreetmap.org and use data munging techniques, such as assessing the quality of the data for validity, accuracy, completeness, consistency and uniformity, to clean the OpenStreetMap data for a part of the world that you care about. Choose to learn SQL or MongoDB and apply your chosen schema to the project.

B - <u>Extract Chosen</u>

I chose the area where live, a suburb of Ottawa (Ontario, Canada), called Kanata. I obtained the extract data by downloading it from https://mapzen.com/data/metro-extracts/ The unzipped file size of the map extract was 115 MB. Below you can find an image from OpenStreetMap which shows the Greater Ottawa Area and the location of my extract area, Kanata. Here is a link to the general area in OSM: http://www.openstreetmap.org/#map=13/45.3159/-75.8985.


Gatineau
Hydrobase Ottawa / Gatineau
Rockcliffe Airport
RR 174
148
Ottawa
417
Constance Lake Waterdome
Connaught Range and Primary Training Centre
416
Aéroport Macdonald-Cartier Airport
7

Note: All code referenced can be found in the submitted scripts generated from Jupyter Notebook, all have a title starting with "Data Wrangling Project".

C - <u>Summary of project process</u>

After downloading the XML (OSM) file, I used the '*Auditing and Cleaning'* script to identify and fix some problems with addressing in the data. I then used the '*Convert XML to Json*' script to take the OSM file and convert its contents to a json file. I then read the file into a MongoDB database table and from there I could run the '*MongoDB Queries*' script on it.

D - <u>Problems encountered in the OSM data for this extract</u>

<u>I wrote python code (and modified code from the OSM Case Study solution) to audit the OSM file and clean/correct the following issues: (outputs are copied and pasted from Jupyter Notebook).</u>

1) <u>All postal codes were not consistent.</u> Proper Postal Codes in Canada follow a L#L #L# format, where # is a number and L is a capital letter. There should be a space between the 3rd and 4th characters in the postal code.

- Correct Postal Codes, Total number: 254, Percentage: 0.92
- Incorrect Postal Codes, Total number: 21, Percentage: 0.08

['K2L1V1','K2T', 'K2K W2W', 'K2L1V1', 'K2L1V1', 'k2S 1B1', 'K2KW2W', 'K2L4E5', 'K2L1V1','K2K2W6', 'K2K2W6', 'K2K2W6', 'K2K2W6', 'k2h 8r8','ON K2K', 'K2L1V1', 'K2L1S2', 'K2L1S2', 'K2K2W6', ''K2K2W6', 'K2K2W6']

From this subset of postal codes, I found 3 reasons why these codes did not comply with the proper format: 1) Not having a space between the 3rd and 4th characters, 2) Not having all caps, or 3) Incomplete codes, that only contained the first 3 characters.

After Programmatic Corrections: (This is just a subset of the results).

```
K2L1V1 => K2L 1V1                K2L4E5 => K2L 4E5
K2T => K2T                       K2L1V1 => K2L 1V1
K2K W2W => K2K W2W               K2K2W6 => K2K 2W6
K2L1V1 => K2L 1V1                K2K2W6 => K2K 2W6
K2L1V1 => K2L 1V1                K2K2W6 => K2K 2W6
K2K W2W => K2K W2W               K2K2W6 => K2K 2W6
```

2) <u>The city name was not consistent.</u> Although there is some disagreement, in general for any part of the greater Ottawa region, the consensus is that the city name should be ‘Ottawa’.

- Correct City Name, Total number: 40601, Percentage: 0.94
- Incorrect City Name, Total number: 2748, Percentage: 0.06

The set of incorrect city names is too long to show here , but all except 3 instances of the incorrect naming were due to the city being called ‘City of Ottawa’, which according to OSM guidelines, is superfluous. The other 3 instances were alternate city names inside of Ottawa, which are used for mailing addresses, but for official purposes, the city should be ‘Ottawa’.

After Programmatic Corrections: (This is just a subset of the results).

City of Ottawa => Ottawa	City of Ottawa => Ottawa
City of Ottawa => Ottawa	Stittsville => Ottawa
City of Ottawa => Ottawa	Stittsville => Ottawa
City of Ottawa => Ottawa	ottawa => Ottawa
City of Ottawa => Ottawa	Kanata => Ottawa
Kanata => Ottawa	Stittsville => Ottawa

3) <u>The province name was not consistent.</u> For the purposes of OpenStreetMap Data, I believe the province should be expressed by its 2 letter code (ON in this case).

- Correct Province Names, Total number: 34, Percentage: 0.6
- Incorrect Province Names, Total number: 23, Percentage: 0.4

['Ontario', 'Ontario', 'Ontario', ‘Ontario', 'Ontario', 'ontario', 'ontario', 'Ontario', 'Ontario', 'ontario', 'Ontario', 'Ontario', 'Ontario', 'Ontario', 'Ontario' 'On', 'Ontario', 'Ontario', 'Ontario', ‘Ontario', 'Ontario', 'Ontario', 'Ontario']

THe most common deviation from the correct ‘ON’ naming was to have the full name of the province, the only other deviation was ‘On’ (not in caps).

After Programmatic Corrections: (This is just a subset of the results).

Ontario => ON	Ontario => ON
Ontario => ON	ontario => ON
Ontario => ON	Ontario => ON
Ontario => ON	On => ON

4) OSM map data is incomplete. When I run the following MongoDB query to see the most frequently occurring postal codes, there is only one postal code with more than 10 occurrences. This suggests to me that there is a lot of missing data in the OSM file, most homes and businesses still aren't included. The one postal code with more than 10 had 87 occurrences, all input by the same user.

```
match = {"$match":{"address.postcode":{"$exists":1}}}
group = {"$group":{"_id":"$address.postcode", "count":{"$sum":1}}}
sort = {"$sort":{"count":-1}}
pipeline = [match,group,sort]
postalcodes = coll.aggregate(pipeline)
```

```
[{u'_id': u'K2T 0K5', u'count': 84},
 {u'_id': u'K2H 8V5', u'count': 9}, ...
```

Note: The problem of street name abbreviations we saw in the case study was not found in the dataset I examined. Street names had already been standardized.

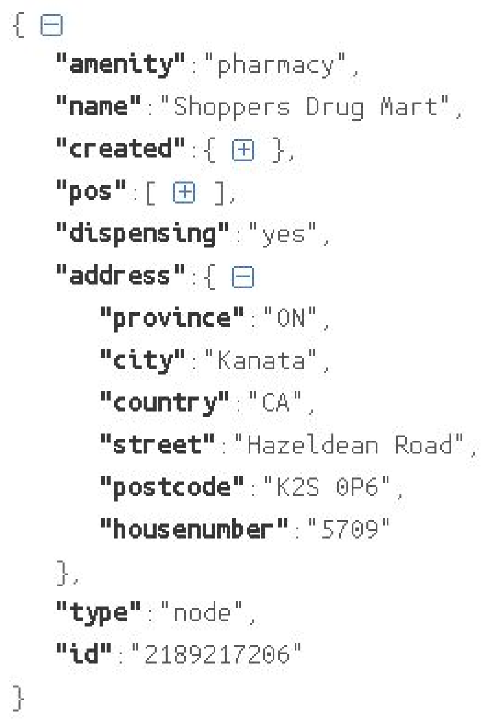
E - Data Overview:

Raw OSM file, tag count:

```
def count_tags(filename):
    tags = {} # initialize diciontary to hold tags
    for event, element in ET.iterparse(filename): # iterate over tags
        if event == 'end': # if closing tab
            if element.tag in tags: # check to see if there is already an entry
                                    #in the dictionary for that tag
                tags[element.tag] += 1 # increase tag counter by 1
        # if 'element.tag' is not a key of tags, add it:
            else:
                tags[element.tag] = 1
    return tags
```

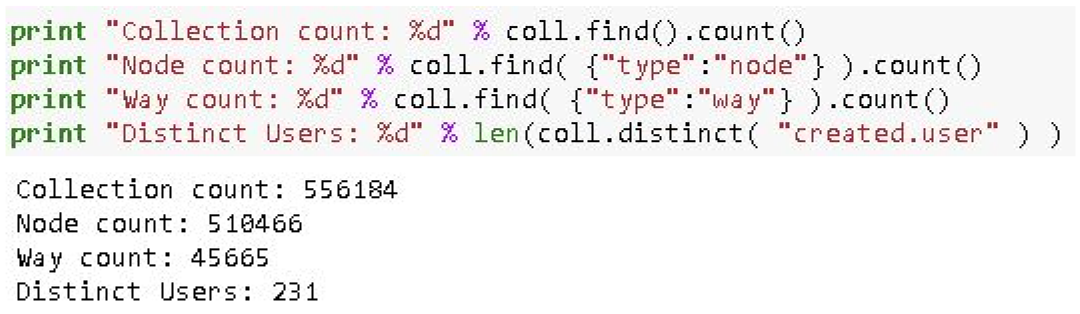
```
{'bounds': 1,
 'member': 3335,
 'nd': 521250,
 'node': 510519,
 'osm': 1,
 'relation': 333,
 'tag': 495019,
 'way': 45665}
```

Example Json node, after conversion from OSM format:



```
{ ⊟
   "amenity":"pharmacy",
   "name":"Shoppers Drug Mart",
   "created":{ ⊞ },
   "pos":[ ⊞ ],
   "dispensing":"yes",
   "address":{ ⊟
      "province":"ON",
      "city":"Kanata",
      "country":"CA",
      "street":"Hazeldean Road",
      "postcode":"K2S 0P6",
      "housenumber":"5709"
   },
   "type":"node",
   "id":"2189217206"
}
```

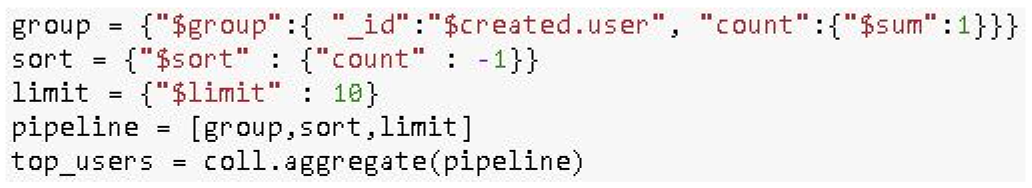
MongoDB Collection Stat Summary:



```
print "Collection count: %d" % coll.find().count()
print "Node count: %d" % coll.find( {"type":"node"} ).count()
print "Way count: %d" % coll.find( {"type":"way"} ).count()
print "Distinct Users: %d" % len(coll.distinct( "created.user" ) )
```

```
Collection count: 556184
Node count: 510466
Way count: 45665
Distinct Users: 231
```

Top Users



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

```
[{u'_id': u'carpbunker', u'count': 338515},
 {u'_id': u'gfysgerm', u'count': 48547},
 {u'_id': u'Johnwhelan', u'count': 27122}, ...
```

Top Amenities

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

```
[{u'_id': u'parking', u'count': 637},
 {u'_id': u'post_box', u'count': 238},
 {u'_id': u'school', u'count': 55},    ...
```

Top Restaurant Cuisine Types

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

```
[{u'_id': None, u'count': 23},
 {u'_id': u'american', u'count': 4},
 {u'_id': u'regional', u'count': 3}, ...
```

Note: The top cuisine type is 'None', which shows that a lot of restaurants are missing data on cuisine type.

Schools by Language

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

```
[{u'_id': u'english', u'count': 28},
 {u'_id': None, u'count': 23},
 {u'_id': u'french', u'count': 4}]
                                      ...
```

F - Additional Ideas

Regarding the inconsistencies in naming, I think each OSM file should come with metadata that contains the agreed upon standards for naming in that particular region. Though it will be hard to reach a consensus on some items, others will not be up for argument and this will help to maintain consistency throughout the OSM data.

Another thing that would cut down on inconsistencies and errors would be to have some form of format validation (only accepts postal codes in proper format, only accepts two character province code etc.)

G - Conclusion

As I have shown, there are problems with naming inconsistencies in the data, but the bigger problem is that the data for this particular area is incomplete, and the most important step to take to improve the data set is to work on completing it by filling in partial and missing data.

The top user query shows that the top user is responsible for 61% of the edits, and the top 3 users combined are responsible for %74 of the edits to the OSM file.

Action must be taken to motivate more people to contribute to OSM and thus start to fill in the missing data. Some ideas for motivating people could be social media campaigns and contests.

H - References

Udacity Sample Data Wrangling Project

Udacity Discussion Forum

https://discussions.udacity.com/

MongoDB Documentation

https://docs.mongodb.com/manual/reference/program/mongoimport/
https://docs.mongodb.com/manual/core/aggregation-pipeline/
https://docs.mongodb.com/manual/reference/method/db.collection.count/

Stack Overflow

https://stackoverflow.com/questions/15171622/mongoimport-of-json-file
https://stackoverflow.com/questions/13145468/mongodb-aggregation-framework-double-match
https://stackoverflow.com/questions/12339251/regex-detect-if-word-contains-lowercase-character
https://stackoverflow.com/questions/717644/regular-expression-that-doesnt-contain-certain-string
https://stackoverflow.com/questions/11648627/mongodb-shells-db-stats-in-php-and-python