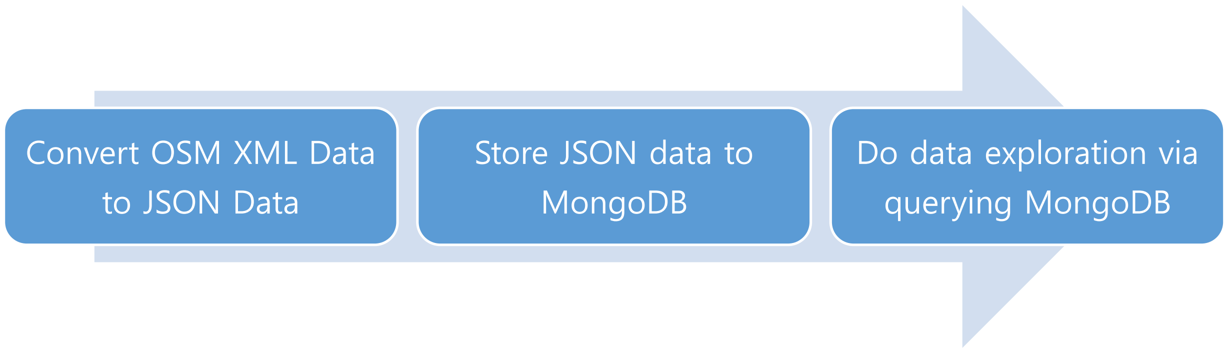
Wrangling Canberra, ACT, Australia OpenStreetMap data!

Author: Jae Hee Lee

1. Brief Process Overview

I have chosen Canberra because this is the place where my University is located in. Furthermore, I tried not to get only the part of Canberra using Overpass API because I really wanted to analyse with large data. However, after importing the large file I sliced the data into a smaller part using MongoDB query.


Convert OSM XML Data to JSON Data
Store JSON data to MongoDB
Do data exploration via querying MongoDB

2. Converting OSM XML data to JSON data

Getting OSM XML Data

I downloaded data from Mapzen. They have already prepared metro extracts for Canberra, ACT, Australia.

Wrangling OSM XML Data

This data cleaning process was probably one of the most difficult parts of this project. Luckily, only addr: key part had problems.

1. **Street Name Issue**

There were a lot of inconsistencies. Here are few examples:

- Grose Street Deakin (Deakin is suburb)
- Tez Automotive (Not a street)
- 12 (just digit nothing really. This can be a house number instead)
- Waramanga Shopping Centre (Name of the shopping centre)
- Thoji Street Unit 4 (Unit 4 should be omitted)

First, I tried to use regular expressions to make all street name consistent but I concluded that it is not possible to simply come up with one regular expression to cover all possibilities. After extracting unique street names, I realized that the variety of street types is fairly limited (i.e. in total there are only 24 possibilities). Thus, I made an exhaustive list of expected street types. To deal with the abbreviations of street types (e.g. St. -> Street). I have mapped them with the full street type names and updated the abbreviated names so that every name has full street type name.

Second, there were few instances where two streets were given within one attr:street key. In this case, I have looked them up and have chosen the main street.

2. Postcode Issue

There were only two errors. Therefore, I have just manually cleaned them (i.e. from 'Yarralumla ACT 2600' to '2600' and from 'ACT 2604' to '2604')

3. City Issue

There were only five errors. Therefore, I have just manually cleaned them (refer to code).

4. Housenumber Issue

The only concern I had was that some of the entries used 'ndash' symbol (which is similar to ascii minus -) Therefore, I had to replace ndash to minus in order to eliminate encoding error. However, in Python 2.7, we cannot replace ndash to minus using replace function as ndash is invalid. As such, I encoded the Unicode character to UTF-8 and then replaced \xe2\x80\x93 with -.

3. Data Overview

File Sizes

- canberra_australia.osm (102,038 KB)
- canberra_australia.osm.json (157,490 KB)

I have created a database called „openstreet" and created a collection called „canberra" and then I have imported canberra_australia.osm.json data and performed some basic queries:

<u>General Information regarding the area selected</u>

Total documents inserted:

db.canberra.find().count() → **548,585 documents**

Total document with the type of node:

db.canberra.find({"type":"node"}).count() → **500,387 documents (91.21%)**

Total documents with the type of way:

db.canberra.find({"type":"way"}).count() → **48,186 documents (8.78%)**

The top –most contributed user:

```
db.canberra.aggregate([{"$group" : {"_id" : "$created.user", "count" : {"$sum" : 1} } },
                        {"$sort" : {"count" : -1}},
                        {"$limit" : 1}])
```
→ **numbfew (90681 documents)**

The number of unique users:

len(db.canberra.distinct("created.user")) → **492**

The number of unique streets:

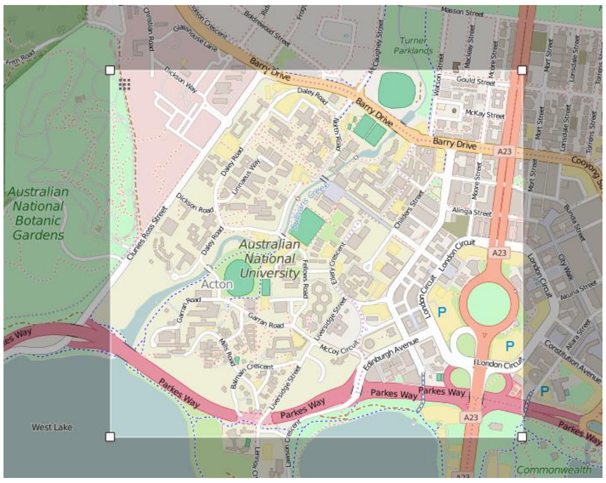
len(db.canberra.distinct("address.street")) → **318**

The number of unique postcodes:

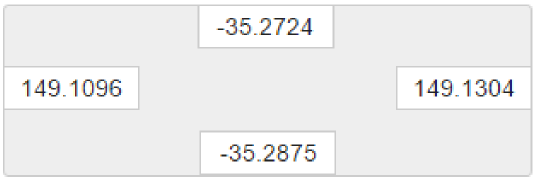
len(db.canberra.distinct("address.postcode")) → **29**

Specific Information

I wanted to see nodes or ways in the area selected below:


Australian National Botanic Gardens
Australian National University
Acton
Turner Parklands
Barry Drive
Parkes Way
West Lake
Commonwealth

The precise position for above area is


-35.2724
149.1096
149.1304
-35.2875

Although I could have used the Overpass API to download a custom square area, I have used the MongoDB query using PyMongo to extract the desired data from the whole dataset.

```
map_query = {    "pos.0" : {"$gte": -35.2875, "$lte":-35.2724},
                 "pos.1" : {"$gte": 149.1096, "$lte": 149.1304} }
map_projection = {"_id" : 0, "created.uid" : 1}
map_contributed_users = db.canberra.find(map_query, map_projection)
```

The number of nodes and ways within this area:

count_specific_area_nodes = map_contributed_users.count() → **8002 (1.46%)**

The number of users who contributed to information about nodes and ways within this area:

```
unique_users = set()
for user in map_contributed_users:
        unique_users.add(user['created']['uid'])
```

len(unique_users) → **89 (18.09%)**

I have lived in Yarralumla and Braddon so I would like to find out information in these suburbs. The postcodes for Yarralumla and Braddon are 2600 and 2612 respectively.

```
postcode_query = {"type" : "node", "address.postcode" : {"$in" : ['2600', '2612']}}
postcode_projection = {"_id" : 0, "created.uid" : 1}
specific_area_nodes = db.canberra.find(postcode_query, postcode_projection)
```

The number of users who contributed to information in 2600 and 2612:

```
unique_users = set()
for user in specific_area_nodes:
    unique_users.add(user['created']['uid'])
```

len(unique_users) → **5 (1.02%)**

The number of nodes and ways in 2600 and 2612:

count_specific_area_nodes = specific_area_nodes.count() → **7 (0.00128%)**

Top 3 biggest religion in Canberra:

```
list(db.canberra.aggregate([{"$match" : {"amenity" : {"$exists": 1} ,
                              "amenity" : "place_of_worship"}},
                              {"$group" : {"_id" : "$religion", "count": {"$sum" : 1}}},
                              {"$sort" : {"count" : -1}},
                              {"$limit" : 3}]))
```
→ **Christian (109), Buddhist (4), Islam (3)**

The 10 amenities in Canberra:

```
list(db.canberra.aggregate([{"$match" : {"amenity" : {"$exists" : 1}}},
                    {"$group" : {"_id" : "$amenity", "count" : {"$sum" : 1}}},
                    {"$sort" : {"count" : -1}},
                    {"$limit" : 10}]))
```
→ **Parking (1154), School (211), Restaurant (204), Drinking Water (192), Toilets (168), Café (158), Bench (145), Bicycle Parking (131), Fast Food (126), Place of Worship (121)**

The top 5 streets in Canberra:

```
list(db.canberra.aggregate([{"$match" : {"address.street" : {"$exists" : 1}}},
                    {"$group" : {"_id" : "$address.street", "count" : {"$sum" : 1}}},
                    {"$sort" : {"count" : -1}}, {"$limit" : 5}]))
```
→ **Enderby Street (49), Beasley Street (37), Delma View (34), The Valley Avenue (30), Henry Kendall Street (26)**

This result does not match the reality because Enderby Street in Mawson is pretty secluded place. Thus, I would now like to question the 'quality' of data.

The question is: **is the data outdated?**

In answering this question, I initially searched for the top 3 frequent dates when these nodes or ways were created and 271 data were created in June 2007, 174 in October 2007 and 113 in May 2008. However, it is not sufficient to say the whole data are outdated with this data because compared to the total number of documents it is almost infinitesimal.

I will regard the 'up-to-date' data as anything that is published after 2013.

The number of documents created between 2014 and 2016 (20 February):

```
time_query = { "created.timestamp" : {"$gte": '2014-01-01T00:00:00Z', "$lte": '2016-02-20T00:00:00Z'}}

time_projection = {"_id" : 0, "created.uid" : 1}

specific_time_nodes = db.canberra.find(time_query, time_projection).count()  →  158528 (28.9%)
```

The number of documents created before 2014:

```
time_query = { "created.timestamp" : {"$lt": '2014-01-01T00:00:00Z'}}

time_projection = {"_id" : 0, "created.uid" : 1}

specific_time_nodes = db.canberra.find(time_query, time_projection).count()  →  390057 (71.1%)
```

Therefore, the data itself are not really reflecting the current situation in my opinion, even though one may claim that I am being too strict. In order to improve the data, more people in Canberra would have to contribute to the OpenStreetMap Canberra data.

Every time I play games, I invest my time primarily to improve the ranking of my character. By applying the same principle to this map contribution 'game', I believe that the data will become more robust. For example, OpenStreetMap and the ACT state government can collaborate by giving good incentives to users who contribute the most. For example, when I was working as an ambassador at Google, I tried to improve the Google Map by hosting Google Map competition and the person who contributed the most received a Nexus smartphone (in fact he contributed a lot!). Without giving a prize, I do not think that people will participate actively. The main reason for this is that ordinary people prefer using Google Map instead of OpenStreetMap. As it is almost good for the state government to invest in this area, I believe that this can be a good opportunity for both parties.

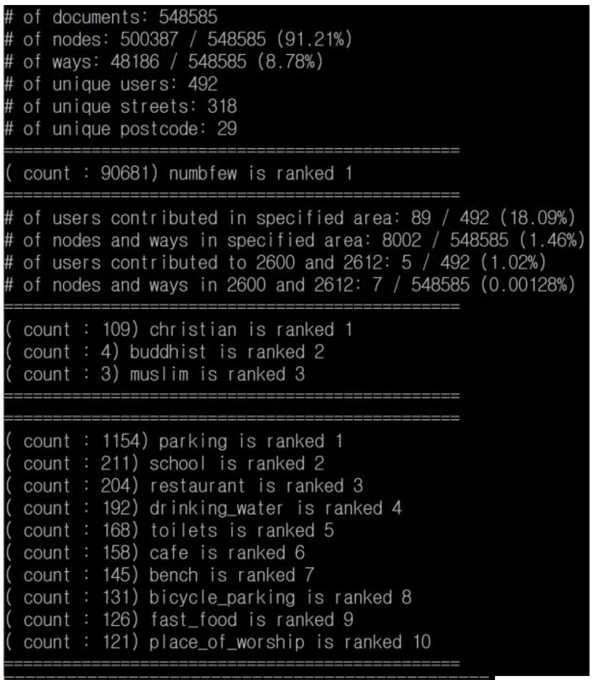
<u>Conclusion</u>

Regardless of the **wrangling** process, as investigated above, the data (i.e. XML OML) is itself **incomplete** and **outdated** to some extent. As such, the exploration above may not necessarily match the reality. For other metrics for assessing data quality, the following points are my conclusion:

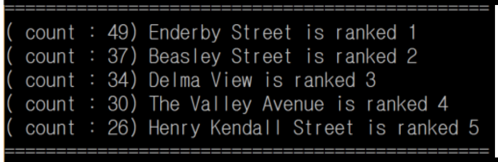
- The data is **valid** because it does conform to the schema.
- The data is **consistent**.
- The data itself is **accurate**.

In order to improve the '**completeness'** of data, there must be some other forms of incentives to attract users from Canberra to more actively contribute to the map.

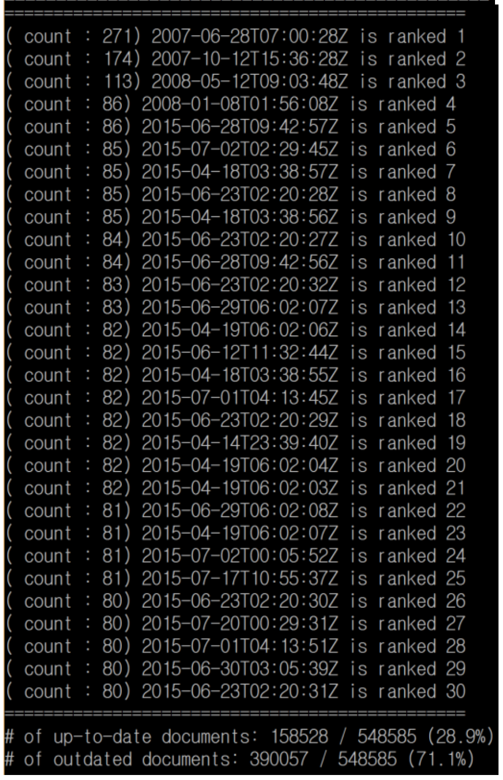
Appendix 1. Query result after running query.py



```
# of documents: 548585
# of nodes: 500387 / 548585 (91.21%)
# of ways: 48186 / 548585 (8.78%)
# of unique users: 492
# of unique streets: 318
# of unique postcode: 29
===============================================
( count : 90681) numbfew is ranked 1
===============================================
# of users contributed in specified area: 89 / 492 (18.09%)
# of nodes and ways in specified area: 8002 / 548585 (1.46%)
# of users contributed to 2600 and 2612: 5 / 492 (1.02%)
# of nodes and ways in 2600 and 2612: 7 / 548585 (0.00128%)
===============================================
( count : 109) christian is ranked 1
( count : 4) buddhist is ranked 2
( count : 3) muslim is ranked 3
===============================================
===============================================
( count : 1154) parking is ranked 1
( count : 211) school is ranked 2
( count : 204) restaurant is ranked 3
( count : 192) drinking_water is ranked 4
( count : 168) toilets is ranked 5
( count : 158) cafe is ranked 6
( count : 145) bench is ranked 7
( count : 131) bicycle_parking is ranked 8
( count : 126) fast_food is ranked 9
( count : 121) place_of_worship is ranked 10
===============================================
```



```
===============================================
( count : 49) Enderby Street is ranked 1
( count : 37) Beasley Street is ranked 2
( count : 34) Delma View is ranked 3
( count : 30) The Valley Avenue is ranked 4
( count : 26) Henry Kendall Street is ranked 5
===============================================
```



```
===============================================
( count : 271) 2007-06-28T07:00:28Z is ranked 1
( count : 174) 2007-10-12T15:36:28Z is ranked 2
( count : 113) 2008-05-12T09:03:48Z is ranked 3
( count : 86) 2008-01-08T01:56:08Z is ranked 4
( count : 86) 2015-06-28T09:42:57Z is ranked 5
( count : 85) 2015-07-02T02:29:45Z is ranked 6
( count : 85) 2015-04-18T03:38:57Z is ranked 7
( count : 85) 2015-06-23T02:20:28Z is ranked 8
( count : 85) 2015-04-18T03:38:56Z is ranked 9
( count : 84) 2015-06-23T02:20:27Z is ranked 10
( count : 84) 2015-06-28T09:42:56Z is ranked 11
( count : 83) 2015-06-23T02:20:32Z is ranked 12
( count : 83) 2015-06-29T06:02:07Z is ranked 13
( count : 82) 2015-04-19T06:02:06Z is ranked 14
( count : 82) 2015-06-12T11:32:44Z is ranked 15
( count : 82) 2015-04-18T03:38:55Z is ranked 16
( count : 82) 2015-07-01T04:13:45Z is ranked 17
( count : 82) 2015-06-23T02:20:29Z is ranked 18
( count : 82) 2015-04-14T23:39:40Z is ranked 19
( count : 82) 2015-04-19T06:02:04Z is ranked 20
( count : 82) 2015-04-19T06:02:03Z is ranked 21
( count : 81) 2015-06-29T06:02:08Z is ranked 22
( count : 81) 2015-04-19T06:02:07Z is ranked 23
( count : 81) 2015-07-02T00:05:52Z is ranked 24
( count : 81) 2015-07-17T10:55:37Z is ranked 25
( count : 80) 2015-06-23T02:20:30Z is ranked 26
( count : 80) 2015-07-20T00:29:31Z is ranked 27
( count : 80) 2015-07-01T04:13:51Z is ranked 28
( count : 80) 2015-06-30T03:05:39Z is ranked 29
( count : 80) 2015-06-23T02:20:31Z is ranked 30
===============================================
# of up-to-date documents: 158528 / 548585 (28.9%)
# of outdated documents: 390057 / 548585 (71.1%)
```