OpenStreetMap Project
Data Wrangling with MongoDB
Mustafa Bilal Tahir

Map Area: Seattle, WA, United States

Overpass API Query for Seattle Area: (node(47.58,-122.39,47.70,-122.26);<;);out meta;

Initial Data Exploration/ Problems Encountered

I processed the data using some Python code to get a sense of it and to see if there were any major issues. The code is all saved in a separate python file.

First, I looked at the structure of the data and parsed through all the tags:

{'member': 11160,
 'meta': 1,
 'nd': 1136464,
 'node': 1044928,
 'note': 1,
 'osm': 1,
 'relation': 932,
 'tag': 834868,
 'way': 114327}

The structure makes sense for an osm file and there are no significant unknown tags. The one tag besides the main tags (nodes, ways and relations) that had a lot of entries is ‘nd’ which is a sub-tag for ways.

Secondly, I went through the data to get a sense of the text being used and to try and isolate any potential problem characters:

lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

So, problem characters would include symbols such as ‘+’ or ‘[‘ or have spaces or new lines etc.

{'lower': 344609, 'lower_colon': 478195, 'other': 12063, 'problemchars': 1}

There was only one potential problem character so the data is looking pretty clean for now. The other category is almost exclusively because of characters with two colons e.g. ‘source:addr:id’.

The last step before loading into MongoDB is to check the street addresses and see if they are all correct and consistent. Most of the data looked good but it was easy to see some abbreviations that were coming through which would be nice to fix. ‘North West’, ‘North East’ , ‘South West’ are frequently abbreviated to NW, N.W., NE, N.E., SW etc.. I added this into the

mapping file to fix the addresses in addition to the standard fixes to Ave -> Avenue, St. -> Street etc.

Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

File sizes

Seattle_reduced.osm ..... 249 MB
Seattle_reduced.json .... 329 MB

Number of documents

```
> db.Seattle_map2.find().count()
1,159,255
```

Number of nodes

```
> db.Seattle_map2.find({"type":"node"}).count()
1,044,928
```

Number of ways

```
> db.Seattle_map2.find({"type":"way"}).count()
114,327
```

Number of unique users

```
> len(db.Seattle_map2.distinct(“created.user”))
600
```

Top 10 contributing users

```
> db.Seattle_map2.aggregate([{"$group":{"_id":"$created.user",
                                        "count":{"$sum":1}}},
                                        {"$sort":{"count":-1}},
                                        {“$limit":10}])

{u'count': 387780, u'_id': u'SeattleImport'}
{u'count': 373819, u'_id': u'Glassman'}
{u'count': 98511, u'_id': u'seattlefyi'}
{u'count': 52377, u'_id': u'jeffmeyer'}
{u'count': 49194, u'_id': u'Foundatron'}
{u'count': 33080, u'_id': u'Sudobangbang'}
{u'count': 23498, u'_id': u'chronomex'}
{u'count': 22799, u'_id': u'sctrojan79'}
{u'count': 11748, u'_id': u's_matthews'}
{u'count': 8335, u'_id': u'STBrenden'}
```

Number of users appearing only once (having 1 post)

```
> db.Seattle_map2.aggregate([{"$group":{"_id":"$created.user",
                                       "count":{"$sum":1}}},
                                       {"$group":{"_id":"$count", "num_users":{"$sum":1}}},
                                       {“$sort":{"_id":1}},
                                       {“$limit":1}])

{u'num_users': 161, u'_id': 1}
```

Top user contribution percentage (“SeattleImport”) - 387780/1159255 = 33.4%
Combined top 2 users' contribution (“SeattleImport” and “Glassman”) - (387780+373819)/1159255 = 65.7%
Similarly, combined Top 10 users contribution - 91.5%
Combined number of users making up only 1% of posts - 551 (about 92% of all users)

Not surprisingly, only a handful of users are responsible for the majority of the entries (top 10 make up over 90% of the data). This makes sense as often the majority of mapping is done via bots (computer programs).

Additional data exploration using MongoDB queries

Top 10 appearing amenities

```
> db.Seattle_map2.aggregate([{"$match":{"amenity":{"$exists":1}}}, {"$group":{"_id":"$amenity",
      "count":{"$sum":1}}}, {“$sort":{"count":-1}}, {"$limit":10}])
{u'count': 2345, u'_id': u'bicycle_parking'}
{u'count': 878, u'_id': u'parking'}
{u'count': 700, u'_id': u'restaurant'}
{u'count': 346, u'_id': u'cafe'}
{u'count': 231, u'_id': u'place_of_worship'}
{u'count': 172, u'_id': u'fast_food'}
{u'count': 158, u'_id': u'bench'}
{u'count': 144, u'_id': u'bar'}
{u'count': 132, u'_id': u'school'}
{u'count': 113, u'_id': u'post_box'}
```

Biggest religion

```
> db.Seattle_map2.aggregate([{“$match”:{"amenity":{"$exists":1},
“amenity”:”place_of_worship"}}, {"$group":{"_id":"$religion", "count":{"$sum":1}}},
 {"$sort":{"count":-1}}, {"$limit":1}])

{u'count': 231, u'_id': None}
```

The reason we are not getting a breakdown is because there is no ‘religion’ category for the entries unfortunately. I made this error while importing the data into json. My script did not process the religion tags and so the ending data does not have them.

The next best thing we can do is to group them by name and see if we can get a sense of what kind of places of worship are predominant.

```
db.Seattle_map2.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"place_of_worship"}},
      {"$group":{"_id":"$name", "count":{"$sum":1}}},
      {"$sort":{"count":-1}}, {"$limit":10}])

{u'count': 4, u'_id': u'Seventh Day Adventist Church'}
{u'count': 3, u'_id': u'The Church of Jesus Christ of Latter-day Saints'}
{u'count': 3, u'_id': None}
{u'count': 3, u'_id': u'Kingdom Hall of Jehovahs Witnesses'}
{u'count': 2, u'_id': u'Bethel Ethiopian Church of Seattle'}
{u'count': 2, u'_id': u'Congregation Beth Shalom'}
{u'count': 2, u'_id': u"Saint Patrick's Church"}
{u'count': 2, u'_id': u'Center for Spiritual Living'}
{u'count': 2, u'_id': u'Tower Memorial Church'}
{u'count': 2, u'_id': u'Calvary Christian Assembly’}
```

As expected, it's mostly Churches suggesting Christianity is the most popular religion. There seems to be a significant Mormon presence as well as the second most popular entry is for LDS Churches.

Most popular cuisines

```
> db.Seattle_map2.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"restaurant"}},
                             {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
                             {"$sort":{"count":-1}}, {"$limit":10}])

{u'count': 200, u'_id': None}
{u'count': 48, u'_id': u'pizza'}
{u'count': 41, u'_id': u'thai'}
{u'count': 41, u'_id': u'mexican'}
{u'count': 34, u'_id': u'italian'}
{u'count': 33, u'_id': u'american'}
{u'count': 32, u'_id': u'chinese'}
{u'count': 22, u'_id': u'japanese'}
{u'count': 21, u'_id': u'vietnamese'}
{u'count': 20, u'_id': u'seafood'}
```

Again, the category 'cuisine' is not populated for a lot of the entries (hence our top hit is 'None'). However, we still get a reasonable amount of entries which are populated so we can still get some insight from this query.

Zip Codes

Seattle Zip Codes should be starting with '98…' and be 5 digits and it seems we have pretty clean data for zip codes for this area as almost all had this feature. The only exception were 3 zip codes starting with 4 e.g. 4139. This could potentially be the zip code being missing and only the sub zip code (the 4 digit extension) being entered. For example one entry is 98109-5210 (zip code followed by 4 digit sub zip code).

Sort postcodes by count, descending

```
db.char.aggregate([{"$match":{"address.postcode":{"$exists":1}}}, {"$group":
{"_id":"$address.postcode", "count":{"$sum":1}}}, {"$sort":{"count":1}}])
```

Here are the top results, beginning with the highest count:

```
{u'count': 17525, u'_id': u'98115'}
{u'count': 16516, u'_id': u'98103'}
{u'count': 10131, u'_id': u'98117'}
{u'count': 9864, u'_id': u'98105'}
{u'count': 8895, u'_id': u'98122'}
….
{u'count': 1, u'_id': u'98109-5210'}
…
{u'count': 1, u'_id': u'4139}
```

Conclusion

Overall, the data looked in good shape. This does not surprise me as I picked an area encompassing downtown Seattle which should be well mapped compared to other areas. However, there is still room for improvement. The addresses in particular can be fixed and standardized. In Seattle, a lot of addresses start or end with ‘North’, ‘South’, ‘North West’ etc. and these are frequently abbreviated to one or two letter short forms.

It was also apparent through some of our queries that some crucial grouping data is missing for some categories. We can improve the data by filling in the group tags for these categories. For example, we can fill in the cuisine tags for the 200 places that have not been given that tag yet (see cuisine query) either manually or running some script that assigns this based on the name of the restaurant.