OSM2MongoDB

Nicholas Sullivan

Map area: Portland OR
https://www.openstreetmap.org/relation/186579#map=11/45.5429/-122.6545

Data:

https://s3.amazonaws.com/metro-extracts.mapzen.com/portland_oregon.osm.bz2

Problems with using the .osm data:

There were several problems I encountered working with the .osm data for the Portland area:

-Memory errors
-Inconsistent street naming conventions
-More than just Portland

Memory errors:

Initially I was using the ElementTree.parse() method for looking at the data set. but as the file size is 583.3 MB which promptly froze my machine and threw some memory errors. I did some research and used the ET.iterprase(). iterparse allowed the scripts to function but they eventually crashed as well. Upon further research I realised I was leaking memory because I was not clearing out the elements as they were loaded into to RAM. so in my final solution for the process I kept track of both start and end events, when I end event occurs the script would clear the element from memory. This keep the memory usage down but I am still leaking memory from somewhere.

Inconsistent street naming conventions:

With any data set it is important to audit and clean data, this is especially true of human generated data. The primary area I focused on was converting abbreviated street names to unabbreviated form. This was accomplished using the auditstreetnames.py script. They script parses the .osm file and builds up a dictionary of unexpected (abbreviated or otherwise nonstandard) street types. This is a sample entry in the dictionary:

```
'Ave': set(['1201 SW 11th Ave',
        '147th Ave',
        '33rd Ave',
        '5th Ave',
        'B Ave',
        'Mississippi Ave',
        'NE 10th Ave',
        …
```

Using the dictionary I built up a mapping dictionary that would be used in the final parsing of the XML to JSON, This is a snippet of the mapping dictionary:

```
mapping = { "St": "Street",
          "St.": "Street",
          "Ave" : "Avenue",
          "Rd." : "Road",
          "Ave." : "Avenue",
          …
```

More than just Portland:

Once the data is successfully into the database, do we really have just the city of Portland? No in fact there are 77 different cities occuring in the data. See:

```
> db.osm.aggregate([{"$match" : {"address.city" : {"$exists" : 1}}}, {"$group" : {"_id" :
"$address.city"}}, {"$group" : {"_id" : null , "count" : {"$sum" : 1}}}])

{ "result" : [ { "_id" : null, "count" : 77 } ], "ok" : 1 }
```

So our data reaches beyond the city limits. Taking a look at the top 10 cities represented (condensed for readability):

```
> db.osm.aggregate([{"$match" : {"address.city" : {"$exists" : 1}}}, {"$group" : {"_id" :
"$address.city", "count" : {"$sum" : 1}}}, {"$sort" : {"count" : -1}}, {"$limit" : 10}])

{ "result" : [{"_id" : "Portland",  "count" : 28868},
          {"_id" : "Beaverton", "count" : 13335},
          {"_id" : "Hillsboro", "count" : 11278},
          {"_id" : "Forest Grove",  "count" : 6090},
          {"_id" : "Canby", "count" : 5135},
          { "_id" : "Cornelius", "count" : 2777},
          { "_id" : "Estacada", "count" : 2184 },
          { "_id" : "Sandy", "count" : 1640 },
          { "_id" : "West Linn", "count" : 1541 },
          { "_id" : "Banks", "count" : 880 }],
   "ok" : 1}
```

So we now know that our dataset includes more than just Portland, so to answer any further questions about the make up of the OSM data about Portland, I will be prefixing my aggreation queries with the following: {“$match” : {“address.city” : “portland”}}.

Data Info:

Size of portland_oregon.osm : 583.3 MB
Size of portland_oregon.osm.json: 639.2 MB

DB stats:

```
> db.osm.stats()
{ "ns" : "portlandOR.osm",
   "count" : 2714044,
   "size" : 710775616,
   "avgObjSize" : 261.88802244915706,
   "storageSize" : 790593536,
   "numExtents" : 18,
   "nindexes" : 1,
   "lastExtentSize" : 207732736,
   "paddingFactor" : 1,
   "systemFlags" : 1,
   "userFlags" : 0,
   "totalIndexSize" : 88071872,
   "indexSizes" : { "_id_" : 88071872 },
   "ok" : 1 }
```

That is 2,714,044 documents taking up 710775616 bytes (.710GB).

Interesting statistics:

Top ten users (condensed output):

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

                 User                                    Number of documents
         "_id" : "Mele Sax-Barnett",                     "count" : 641656
         "_id" : "Peter Dobratz_pdxbuildings",           “count" : 368409
         "_id" : "Grant Humphries",                      "count" : 354783
         "_id" : "Darrell_pdxbuildings",                 "count" : 316618
         "_id" : "baradam",                              "count" : 143717
         "_id" : "Peter Dobratz",                        "count" : 123505
         "_id" : "amillar-osm-import",           "count" : 112061
         "_id" : "woodpeck_fixbot",                      "count" : 97205
         "_id" : "Paul Johnson",                         "count" : 53096
         "_id" : "tchaddad_imports",                     "count" : 49040
```

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

                  "_id" : "toptenusers",
                    "count" : 2260090
```

So the top ten users account for 83.27% of total documents, really the top contributors make up a huge majority of documents. You can see that their average is more than 82 times larger than the average user (condensed for readability).

```
> db.osm.aggregate([{"$match" : {"created.user" : {"$exists" : 1}}}, {"$group" : {"_id" :
"$created.user" , "count" : {"$sum" : 1}}}, {"$sort" : {"count" : -1}}, {"$limit" : 10}, {"$group" :
{"_id" : "toptenavg", "avg" : {"$avg" : "$count"}}}])

          "_id" : "toptenavg",          "avg" : 226009
```

```
> db.osm.aggregate([{"$match" : {"created.user" : {"$exists" : 1}}}, {"$group" : {"_id" :
"$created.user" , "count" : {"$sum" : 1}}}, {"$sort" : {"count" : -1}}, {"$group" : {"_id" : "alluseravg",
"avg" : {"$avg" : "$count"}}}])
          "_id" : "alluseravg",         "avg" : 2752.580121703854
```

Other queries:

Top ten popular types of amenities (condensed for readability):

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

        "_id" : "parking",              "count" : 2198
        "_id" : "place_of_worship",     "count" : 1152
        "_id" : "restaurant",           "count" : 1002
        "_id" : "school",               "count" : 894
        "_id" : "fast_food",            "count" : 481
        "_id" : "cafe",         “count" : 356
        "_id" : "fuel",         "count" : 291
        "_id" : "bicycle_parking",      "count" : 253
        "_id" : "post_box",             "count" : 227
        "_id" : "bank",                 "count" : 220
```

This seems to be fairly standard, except I suspect most places don’t have cafes or bike parking as top amenities.

Top restaurant types (condensed for readability):

```
> db.osm.aggregate([{"$match" : {"amenity" : {"$exists" : 1}, "amenity" : "restaurant"}}, {"$group" :
{"_id" : "$cuisine" , "count" : {"$sum" : 1}}}, {"$sort" : {"count" : -1}}, {"$limit" : 5}])

          "_id" : null,                 "count" : 357
          "_id" : "pizza",              "count" : 89
```

```
"_id" : "mexican",           "count" : 82
"_id" : "american",          "count" : 60
"_id" : "chinese",           "count" : 40
```

This is odd, I wonder what kind of restaurants are null type. Lets look into this (condensed for readability).

```
> db.osm.aggregate([{"$match" : {"amenity" : {"$exists" : 1}, "amenity" : "restaurant", "cuisine" :
null}}, {"$group" : {"_id" : "$name"}}, {"$limit" : 5}])
```

```
"_id" : "Sunset Grill"
"_id" : "Flying Elephants Delicatessen"
"_id" : "Aquariva"
"_id" : "Cheese & Crack"
"_id" : "Jolly Roger"
```

I looked at the full list and it just seems to be uncategorized restaurants, nothing nefarious. In the future these null values should be addressed with auditing and cleaning.

Conclusions and future efforts:

The initial data retrieved from OSM clearly includes more than just the city of portland, it has entries from 77 cities in the portland area. There are inconsistencies with street naming conventions which were addressed and there are also uncategorized. The finding of the null values for restaurant cuisine types leads me to believe that there are other fields that may be unlabeled and future efforts should take a look into possible ways to audit and clean the lack of fields. Also for future efforts, given the number of other cities represented in the dataset, it might be interesting to take the GPS position and city of each element and plot them differentiating by city, then overlay that plot on a map of the area for getting a sense of the region.