OpenStreetMap Project

Data Wrangling with MongoDB

Bhavya Garg

Map Area: DC-Baltimore

**INTRODUCTION**

I chose this particular area because it is my neighbor city. This domain knowledge was very helpful as I could clean the data with some context instead of doing guesswork or doing extensive research on another city. I chose the DC-Baltimore, instead of the Richmond city from where i belong to because of the limited data that has been populated so far for Richmond on OpenStreetMaps and I needed to analyze a dataset larger than 50 MB for this project. Below is the list of .py files which shows my full work of this project.

**Section 1: Problems Encountered in the Map**

**Unexpected Tags**

mapparser.py was used to count occurrences of each tag, with a result:

* member : 4765
* nd : 488617
* node : 418014
* osm : 1
* relation : 335
* tag : 251958
* way : 40621

Then in Users.py, I find out the number of unique users who have contributed to the map in this particular area!

**Street type abbreviations standardized**

I changed many street types to a non-abbreviated form after discovering these issues in the audit (before importing into MongoDB). For example, below are the 3 types of street which I standardized by adding them to the mapping dictionary in the audit file (Audit.py).

'St': 'Street', 'St.': 'Street', 'ST': 'Street'

Other than Street below are the list of changes i made to the variable mapping :

* "Ave": "Avenue",
* "NW": "North West",
* "E": "East",
* "E.":"East",
* "W.":"West",
* "N.":"North",
* "S.":"South",
* "West":"West",
* "Northeast":"North East",
* "SE" :"South East",
* "Blvd": "Boulevard",
* "Blvd.":"Boulevard",
* "CIrcle":"Circle",
* "Hwy": "Highway",
* "Rd.": "Road",
* "St.":"Street",
* "St" :"Street",
* "ST": "Street",
* "Rdt" :"Road",
* "Ln" : "Lane",
* "SW" : "South West",
* "Ct": "Court",
* "Cir": "Circle",
* "AVE": "Avenue"

**Section 2: Data Overview**

This section contains basic statistics about the dataset and the MongoDB queries used to gather them. The queries are included in query.py.

File sizes:

> coll.dataSize()

329734167

* sample.osm: 90.3 MB
* sample.osm.json: 100 MB

Number of documents:

> coll.count()

1375905

Number of nodes and ways:

> coll.find({'type':'node'}).count()

1254042

> coll.find({'type':'way'}).count()

121863

Number of unique users:

coll.distinct("created.user").length

1194

Top 10 contributing user:

> coll.aggregate([{"$group":{ "\_id":"$created.user", "count":{"$sum":1}}},{"$sort":{"count":-1}},{"$limit":10}]).pretty()

{ "\_id" : "woodpeck\_fixbot", "count" : 284193 }

{ "\_id" : "asciiphil", "count" : 262362 }

{ "\_id" : "aude", "count" : 194694 }

{ "\_id" : "kriscarle", "count" : 50322 }

{ "\_id" : "mdroads", "count" : 47718 }

{ "\_id" : "wonderchook", "count" : 40755 }

{ "\_id" : "RJCorazza", "count" : 33381 }

{ "\_id" : "JoshD", "count" : 32724 }

{ "\_id" : "Evanator", "count" : 30183 }

{ "\_id" : "Your Village Maps", "count" : 29133 }

Number of users contributing only once:

> coll.aggregate([{

... ... '$group': {

... ... '\_id': '$created.user',

... ... 'count': {

... ... '$sum': 1

... ... }

... ... }

... ... }, {

... ... '$group': {

... ... '\_id': '$count',

... ... 'num\_users': {

... ... '$sum': 1

... ... }

... ... }

... ... }, {

... ... '$sort': {

... ... '\_id': 1

... ... }

... ... }, {

... ... '$limit': 1

... ... }])

{ "\_id" : 3, "num\_users" : 327 }

Most common street address:

> coll.aggregate([{

... ... '$match': {

... ... 'address.street': {

... ... '$exists': 1

... ... }

... ... }

... ... }, {

... ... '$group': {

... ... '\_id': '$address.street',

... ... 'count': {

... ... '$sum': 1

... ... }

... ... }

... ... }, {

... ... '$sort': {

... ... 'count': -1

... ... }

... ... }, {

... ... '$limit': 1

... ... }])

{ "\_id" : "13th Street NW", "count" : 135 }

Sort cities by count, descending

> coll.aggregate([{

... ... ... '$match': {

... ... ... 'address.city': {

... ... ... '$exists': 1

... ... ... }

... ... ... }

... ... ... }, {

... ... ... '$group': {

... ... ... '\_id': '$address.city',

... ... ... 'count': {

... ... ... '$sum': 1

... ... ... }

... ... ... }

... ... ... }, {

... ... ... '$sort': {

... ... ... 'count': -1

... ... ... }

... ... ... }])

{ "\_id" : "Leesburg", "count" : 411 }

{ "\_id" : "UNIVERSITY PARK", "count" : 186 }

{ "\_id" : "Alexandria", "count" : 108 }

{ "\_id" : "Baltimore", "count" : 81 }

{ "\_id" : "Sterling", "count" : 72 }

{ "\_id" : "Towson", "count" : 51 }

{ "\_id" : "Frederick", "count" : 51 }

{ "\_id" : "Herndon", "count" : 42 }

{ "\_id" : "Arlington", "count" : 24 }

{ "\_id" : "Silver Spring", "count" : 21 }

{ "\_id" : "Cockeysville", "count" : 21 }

{ "\_id" : "Falls Church", "count" : 18 }

{ "\_id" : "Washington", "count" : 18 }

{ "\_id" : "Pikesville", "count" : 15 }

{ "\_id" : "Columbia", "count" : 15 }

{ "\_id" : "Randallstown", "count" : 15 }

{ "\_id" : "Deale", "count" : 12 }

{ "\_id" : "Churchton", "count" : 12 }

{ "\_id" : "Washington, DC", "count" : 12 }

{ "\_id" : "Kensington", "count" : 12 }

Nodes without addresses:

> coll.aggregate([{

... ... '$match': {

... ... 'type': 'node',

... ... 'address': {

... ... '$exists': 0

... ... }

... ... }

... ... }, {

... ... '$group': {

... ... '\_id': 'Nodes without addresses',

... ... 'count': {

... ... '$sum': 1

... ... }

... ... }

... ... }])

{ "\_id" : "Nodes without addresses", "count" : 1249698 }

Sort postcodes by count, descending

> coll.aggregate([{

... ... ... '$match': {

... ... ... 'address.postcode': {

... ... ... '$exists': 1

... ... ... }

... ... ... }

... ... ... }, {

... ... ... '$group': {

... ... ... '\_id': '$address.postcode',

... ... ... 'count': {

... ... ... '$sum': 1

... ... ... }

... ... ... }

... ... ... }, {

... ... ... '$sort': {

... ... ... 'count': -1

... ... ... }

... ... ... }])

{ "\_id" : "20176", "count" : 321 }

{ "\_id" : "21043", "count" : 279 }

{ "\_id" : "20782", "count" : 186 }

{ "\_id" : "21228", "count" : 138 }

{ "\_id" : "21093", "count" : 96 }

{ "\_id" : "22314", "count" : 96 }

{ "\_id" : "20175", "count" : 96 }

{ "\_id" : "21042", "count" : 90 }

{ "\_id" : "21204", "count" : 66 }

{ "\_id" : "21286", "count" : 54 }

{ "\_id" : "21703", "count" : 51 }

{ "\_id" : "20164", "count" : 42 }

{ "\_id" : "20171", "count" : 42 }

{ "\_id" : "21227", "count" : 36 }

{ "\_id" : "21208", "count" : 36 }

{ "\_id" : "21045", "count" : 33 }

{ "\_id" : "21224", "count" : 30 }

{ "\_id" : "20147", "count" : 30 }

{ "\_id" : "21146", "count" : 30 }

{ "\_id" : "21244", "count" : 27 }

**Conclusion**

There are still several opportunities for cleaning and validation that I left unexplored. Of note, the data set is populated only from one source: OpenStreetMaps. While this crowdsourced repository pulls from multiple sources, some of data is potentially outdated.