OpenStreetMap Project for Data Wrangling with MongoDB

Li Liang

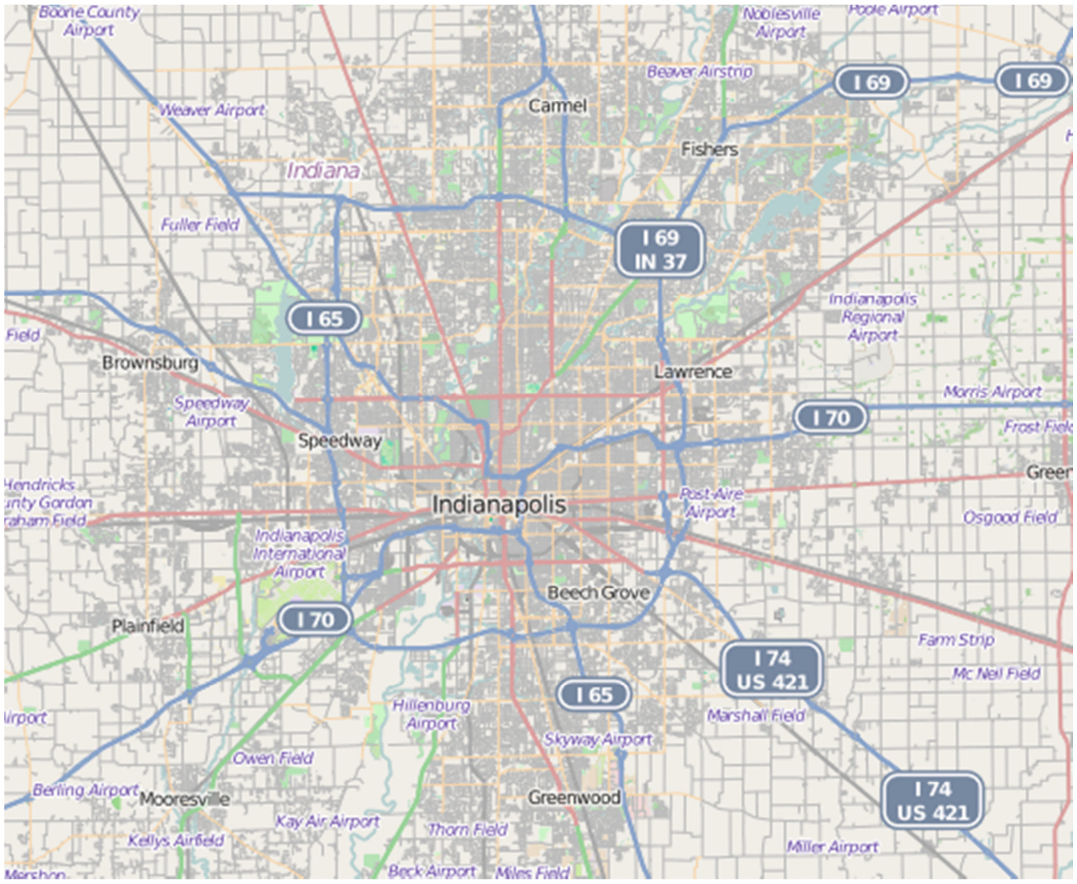
Map Area: Indianapolis, Indiana, United States

I chose this area because I attended high school in Carmel.

Map Zen:

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

https://www.openstreetmap.org/#map=10/39.7948/-86.3553



Contents:

1. Problems Encountered in the Map

Street names abbreviated

Zip codes longer (8 digits vs. 5 digits)

Phone numbers in various formats

2. Data Overview

3. Additional Ideas

Contributor statistics and gamification suggestion

Additional data exploration using MongoDB

4. Conclusion

5. References

1. Problems Encountered in the Map

Using a modified version of data.py, I discovered there were a few street names abbreviations unaccounted for and I added them to **mapping** array. Also, a few new valid regional names such as Circle, Crossing, Pass et al. were added to the **expected** array.

Zip codes I found were not consistent. A majority of them were 5 digits but a few were extended (8 digit) so I used a regex to remove the add-on code so we can aggregate easier in MongoDB.

# Sort zip codes by count, descending

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

Here are the top 5 most frequent zip codes:

```
{ "_id" : "46112", "count" : 111 }
{ "_id" : "46038", "count" : 49 }
{ "_id" : "46256", "count" : 35 }
{ "_id" : "46240", "count" : 32 }
{ "_id" : "46250", "count" : 23 }
```

Three out of the top 5 are in Castleton (46250, 46240, 46256), which is a densely populated area in the north, northwest of Indianapolis. It is close to many lakes and also has a large retail shopping mall, where I used to work. The second most frequent zip code is Fishers, IN (46038) a northwest suburb. The most used zip code is Brownsburg, IN (46112) a suburb west of the city. Before the regex, this would have been more difficult to produce.

Phone numbers were also in a variety of formats. I Googled a regex online[1] and applied it to the **phone** field.

2. Data Overview

This section contains statistical summary of the dataset and the respective MongoDB queries used.

File sizes

```
indy.osm ......... 268 MB
indy.osm.json .... 300 MB
```

# Number of documents

```
> db.indy.find().count()
1356221
```

# Number of nodes

```
> db.indy.find({"type":"node"}).count()
1231835
```

```
# Number of ways
> db.indy.find({"type":"way"}).count()
124386

# Number of unique users
> db.indy.distinct("created.user").length
552

# Top 5 contributing users
> db.indy.aggregate( [{"$group" : {"_id":"$created.user",  "count": {"$sum":1} }},
                              {“$sort":{"count":-1}},
                              {"$limit":5} ] )

{ "_id" : "woodpeck_fixbot", "count" : 401279 }  33%
{ "_id" : "rama_ge", "count" : 360319 }  29%
{ "_id" : "debutterfly", "count" : 154934 }  13%
{ "_id" : "Dr Centerline", "count" : 75155 }  6%
{ "_id" : "svance92", "count" : 62162 }  5%

# Number of users appearing only once (having 1 post)
> db.indy.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                           {"$group":{"_id":"$count", "num_users":{"$sum":1}}},
                           {"$sort":{"_id":1}}, {"$limit":1}])

{ "_id" : 1, "num_users" : 103 }
# “_id” represents postcount
```

3. Additional Ideas

Similar to the sample project, it seems the contribution of data has been skewed towards a few users. The top 5 users comprise 85% of all documents in this collection. This leads me to conclude that this open source tool hasn’t crowd sourced a lot of its data but done so via select users or bots. “Gamification” could help but the Google maps product has more information and a richer dataset. This is a useful tool to explore and study with but as a viable tool maybe they could focus on a particular segment of their users. For example, they could gear information to find public schools, day care centers, parking lots.

Additional data exploration using MongoDB queries

```
# Top 5 appearing amenities
> db.indy.aggregate([{"$match":{"amenity":{"$exists":1}}},
                              {"$group":{"_id":"$amenity",
                              "count":{"$sum":1}}},
                              {"$sort":{"count":-1}},
                              {"$limit":5}])
```

```
{ "_id" : "parking", "count" : 6981 }
{ "_id" : "place_of_worship", "count" : 756 }
{ "_id" : "school", "count" : 513 }
{ "_id" : "grave_yard", "count" : 326 }
{ "_id" : "restaurant", "count" : 231 }
```

# Biggest religion

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

{ "_id" : "christian", "count" : 721 }
```

# Most popular cuisines

```
> db.indy.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"restaurant"}},
                        {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
                        {"$sort":{"count":-1}}, {"$limit":2}])
 { "_id" : null, "count" : 93 }
{ "_id" : "pizza", "count" : 22 }
```

A closer look at the restaurants with null cuisine reveals they are mainly chain restaurants.

```
db.indy.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"restaurant", "cuisine": null}},
                        {"$project" : {"_id" :"$name"}}])

{ "_id" : "Hard Rock Cafe" }
{ "_id" : "TGI Fridays" }
{ "_id" : "Steak 'n Shake" }
{ "_id" : "Donato's" }
{ "_id" : "The Cheesecake Factory" }
{ "_id" : "Jersey Mike's Subs" }
{ "_id" : "Panera Bread" }
{ "_id" : "Noodles & Company" }
```

4. Conclusion

The OSM dataset has been fun to play with and explore. The Indianapolis data is pretty clean but has obvious gaps. I chose NYC before that because that's where I live and I found that the Map Zen data set was too large to process for my computer (over 2 GB) and one I selected from Overpass API, although more manageable(Manhattan, BK, Queens only), was mostly done by scripts or bots. I guess NYers can't be bothered. Overall, I enjoyed this project and hope to explore more datasets.

5. References

1. Regex deal with phone numbers

http://www.diveintopython.net/regular_expressions/phone_numbers.html

http://www.tutorialspoint.com/python/python_reg_expressions.htm