OpenStreetMap Project
Data Wrangling with MongoDB // ND002
Andrew Trick

Map Area: Columbus, OH. United States
https://www.openstreetmap.org/relation/182706#map=11/39.9979/-82.9588

1. Problems Encountered in the Map

After downloading the Columbus.osm dataset from OpenStreetMap and giving it an initial look over, I noticed three problems with the data. I will discuss in the following order:

- Steet abbreviations need consistency ('Street', 'St.', 'st', etc)
- Postal codes lack uniformity ('34215', 'Westerville OH 43082', etc)
- Start date fields missing

Street name abbreviation consistency

Initial analysis of the dataset provided me with the information needed to correct the inconsistencies of street name abbreviations. In the process of converting the .osm to .json, the code included a function to check for the street abbreviation substring. The code then compared every abbr. to a dictionary I prepared from the original dataset and fixed the problem. // (ex: 'Indianola Avenue', Fairwood Ave.' > 'Indianola Ave', Fairwood Ave')

Postal code uniformity

Several of the postal codes, which for the most part followed a strict five digit number, were lacking the uniformity needed as their input. Some had extra or fewer digits, while others included 'OH' and city names in their field. After querying in MongoDB and finding the problem fields, I updated the documents to the correct input if one was determinable.
// (ex: 'Westerville OH 43082' > '43082)

#Postal code query

```
     >db.cbus.aggregate([{'$match': {'address.postcode':
{'$exists': 1}}},{'$group' :{'_id': '$address.postcode',
```

```
'name':{'$push':'$name'},'count': {'$sum': 1}}},{'$sort':
{'count':-1}}])
```

{u'_id': u'43220-4800', u'count': 1, u'name': [u'The Wellington School']}, etc..

Due to the small amount of incorrect postal fields, I decided to include the name in the query and then manually set the update to the document using the name and new postal code. This allowed for a flexibility in the updating that wouldn't be available for a larger problem area.

```
#Postal code update example
>db.cbus.update({'name': 'The Wellington School'},
 {'$set': {'address.postcode': '43220'}})
```

Missing start dates

One of my primary interest in this dataset was to determine if OSM was able to stay current and keep up to date with a large amount of businesses starting up in the Columbus area. Unfortunately, the start_date field is almost non-existent: Out of 341 recored shops located in Columbus, only 2 have a listed start dates.

```
#Start date count query
>len(db.cbus.distinct('start_date'))
 2
```

The only way to solve this problem is to query for a list of every shop without the start_date field and research the information to manually provide and update it onto OpenStreetMap. Possibly finding a 3rd party site (ex: yellowpages.com) and using a bot to scrape and update the information is the only realistic solution. I did not take this step though and had to do without correcting this field.

2. Data Overview

A basic statistics overview of the dataset and the MongoDB queries used to gather them.

File Sizes

Columbus.osm ..... 89 MB
Columbus.osm.json ..... 97 MB

```
#Number of  Documents
```

```
> db.cbus.find().count()
414361
```

#Number of Nodes

```
>db.cbus.find({'type': 'node'}).count()
371147
```

#Number of Ways

```
>db.cbus.find({'type': 'way'}).count()
43208
```

#Number of Unique Users

```
>len(db.cbus.distinct('created.user'))
505
```

#Number of Shops

```
>db.cbus.find({'shop':{'$exists': 1}}).count()
341
```

#Number of Cities (Columbus Metropolis Area)

```
>len(db.cbus.distinct('address.city'))
28
```

3. Additional Ideas

Discovering prime areas to open a business

The ability to query the Columbus OSM data with MongoDB could allow someone with very little knowledge of the city find an opportune place to open a business or shop.
For example, if someone was looking to open a coffee shop near other walk up shops (clothing, bookstore, etc) , they could discover:

Number of cafes successful in the city area: 31
Post code with most cafes: 43212
Street with most shops: Neil Ave
Post codes with most walk up shops (tied) : 43235, 43212

Looking at the results of our queries, our example coffee enthusiest could determine that the 43235 post code area is the best place to begin looking into opening a shop. While it is tied for most walk up shops, the 43212 area already has several established cafes in business to compete against.

The problem with this data, (aside from the overall lack of address information), is similar to problem 3 stated above; missing start date fields. I believe if one were able to query more in depth, (using start date, grouped by postal code/street,) they could discover trends in the recent years and find specific areas in the city that saw large growth succeed.

Additional data exploration using MongoDB queries

#Top 10 cuisine types

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

[{u'_id': u'burger', u'count': 44},
{u'_id': u'mexican', u'count': 22}, ...

#Coffee shops and bookstores grouped via postal code

```
>db.cbus.aggregate([{'$match': {'$or': [{'$and':
[{'amenity' : 'cafe'},{'address.postcode':
{'$exists': 1}}]},{'$and': [{'shop' : 'books'},
{'address.postcode': {'$exists': 1}}]}]}},
{'$project': {'_id': '$address.postcode','coffee':
{'$cond': [{'$eq': ['$amenity', 'cafe']}, '$name',
'$null']},'books': {'$cond': [{'$eq': ['$shop',
'books']}, '$name', '$null']}}},{'$group':{'_id':
'$_id','coffee': {'$push': '$coffee'},'books':
{'$push': '$books'}}}])
```

{u'_id': u'43212',
u'books': [u'Barnes and Noble', u'Acorn Bookstore'],
u'coffee': [u"Luck Bros' Coffee House", u"Cup O' Joe", u'Starbucks']}, ...

#Number of Places of Worship per Religion

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

[{u'_id': u'christian', u'count': 871},
{u'_id': None, u'count': 47},
{u'_id': u'jewish', u'count': 5},
{u'_id': u'muslim', u'count': 4},
{u'_id': u'scientologist', u'count': 1},
{u'_id': u'unitarian_universalist', u'count': 1}]}

#List of Aikido Dojos w/ Address (none unfortunatly)

```
>db.cbus.aggregate([{'$match': {'$or' : [{'sport' :
'aikido'},{'amenity': 'dojo'}]}},{'$group': {'_id':
'$name','number': {'$push':'$address.housenumber'},
'street':{'$push':'$address.street'}}}])
```

u'result': []

Conclusion

After browsing, cleaning, and analysing this dataset with MongoDB, it is apparent that there are many flaws with the OSM data for Columbus Oh. The primary factors of which are consistancy, missing field entries, and keeping the data current. With the majority of information already coming to OSM via bots, I believe it would be possible to update these to include a search to find the missing fields of already entered nodes, while also scraping the data needed to add any new businesses or shops to stay current.