OpenStreetMap Project
Data Wrangling with MongoDB

Yunxiao Wang

Map Area: Richmond, Virginia, United States

1. Problems Encountered in Cleaning the Data

After auditing the map data, I noticed several problems with the data:

- Over-abbreviated street names(such as “Pky” for “Parkway”, “S” for “South).
- Not every word starts with a capital letter(“south linden street”)
- Cities other than “Richmond” is also included
- Inconsistent city name for “Richmond”(“Richmond” and “Richmond City”)
- Postcodes have different length(“23103”, “23227-1107”)
- Postcodes don’t all start with the same area codes(“23223” and “843050”)
- The existing “type” tags in the dataset can overwrite the “node” or “way” “type” fields in the updated data.

Different City Names

Here are the city names appeared and the number of times they appeared:

{'Richmond City': 1, 'Tuckahoe': 1, 'Bon Air': 3, 'Downingtown ': 1, 'Henrico': 14, 'Sandston': 1, 'Manakin Sabot': 1, 'Richmond': 434, 'Mechanicsville': 4, 'Midlothian': 12, 'Glen Allen': 69, 'glen Allen': 1, 'Chesterfield': 4, 'North Chesterfield': 3}

After looking into each city, it’s clear that Downingtown is the only place not in the Richmond area. So this dataset is not just Richmond itself but rather the Richmond area. However Downingtown should not be included as it’s in fact in a different state PA. I found the coordinates of Downingtown have been mistakenly recorded as close to Richmond while the other information of Downingtown seems to be accurate. I decided to remove it from the dataset. I also changed the single occurrence “Richmond City” to “Richmond” for consistency.

Postcodes

I stripped all 4-digit zip code extensions after “-“ or “\”, leaving only the 5-digit zip codes at the beginning.

Three of the zip codes do not start with “23”, one of them is Downingtown which I’ve already decided to delete. After following the links to websites in the other two documents, I realized the user might have mistakenly used the P.O. box number as

post codes. I changed the wrong zip codes to the correct ones provided by the websites.

Street Names

After examining all street names, there're more street types than I originally expected. The complete street type list after including those unexpected is:
["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane", "Road", "Trail", "Parkway", "Commons", "Alley", "Loop", "Way", "Turnpike", "Circle", "Highway", "West"]

Some of the street types were abbreviated, I updated all of them to the full name. I also noticed not only street types were abbreviated, for example, "S." was used instead of "South". To find all abbreviations no matter if they're at the end of the street names, I used the regular expression:

```
bad_names =
re.compile(r'Pky\b\.?|Pkwy\b\.?|Rd\b\.?|street\b\.?|Ave\b\.?|St\b\.?|Ct\b\.?|E\b\.?|S\b\.?|W\b\.?|N\b\.?|NE\b\.?|SE\b\.?|SW\b\.?|NW\b\.?')
```

The code I used to fix the over-abbreviated names and make sure every word starts with a capital letter is(change "S. Addison St" to "South Addison Street" and "south linden street" to "South Linden Street"):

```
street = line['address']['street']
# first make sure every word in street starts with capital letter
street_list = street.split(' ')
street_list = [word.capitalize() for word in street_list]
street = ' '.join(street_list)
# mapping is a dictionary that maps abbreviation to full name
m = bad_names.search(street)
while m:
    start = m.start(0)
    end = m.end(0)
    bad = m.group()
    good = mapping[bad[:-1] if bad[-1] == '.' else bad]
    street = street[:start] + good + street[end:]
    m = bad_names.search(street)
line['address']['street'] = street
```

"type" Tags:

After fixing all other problems, I did a sanity check and quickly found out some of the nodes and ways already had a key named "type", which replaced the "node" or "way" "type" in the result. For instance, a "way" element has a "type" key and the corresponding value is "Trail". I made a decision to rename the already existing "type" keys to:

```
element.tag + '_type"
```

2. Data Overview

Here're some basic mongodb queries I ran against the dataset and the statistics I gathered.

File sizes

richmond_virginia.osm 115.2MB
richmond_virginia.json 127.1MB

 # Number of documents
> db.richmond.find().count()
562366

 # Number of nodes
>db.richmond.find({"type":"node"}).count()
514527

 # Number of ways
>db.richmond.find({"type":"way"}).count()
47839

 # Number of unique users(by "user")
>db.richmond.distinct("created.user").length
322

 # Number of unique users(by "uid")
>db.richmond.distinct("created.uid").length
322

 # Top 5 contributing user
> db.richmond.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum": 1}}},
{"$sort":{"count":-1}}, {"$limit":5}])
{ "_id" : "woodpeck_fixbot", "count" : 261901 }
{ "_id" : "Tej Kanitkar", "count" : 58213 }
{ "_id" : "lBluPhoenixl", "count" : 56006 }
{ "_id" : "Omnific", "count" : 41597 }
{ "_id" : "gpstrails", "count" : 24789 }

 # Number of users who have only 1 post
> db.richmond.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum": 1}}},
{"$match": {"count": 1}},{"$group":{"_id":"$count", "num_users":{"$sum":1}}}])

{ "_id" : 1, "num_users" : 70 }

3. Additional Ideas

As it is the case in the sample project, the contributions of users is also very skewed, the top contributor "woodpeck_fixbot" happens to be the second contributor in the sample project. The contribution percentage of "woodpeck_fixbot" is 46.57%, which is almost 5 times as much as the second contributor. The combined top 5 users contribution is 78.69%. This is possibly due to the use of map editing bot versus human editors. It is very likely the top use whose name contains the word "bot", is indeed a bot.

 # The Frequency Users Updated The Data

> db.richmond.aggregate([{"$group":{"_id": "$created.version", "count":{"$sum":1}}}, {"$sort":{"count":-1}},{"$limit":5}])
{ "_id" : "2", "count" : 297213 }
{ "_id" : "1", "count" : 232759 }
{ "_id" : "3", "count" : 20518 }
{ "_id" : "4", "count" : 5902 }
{ "_id" : "5", "count" : 2538 }

> db.richmond.aggregate([{"$group": {"_id": "max_version", "max_version": {"$max": "$created.version"}}}])
{ "_id" : "max_version", "max_version" : "9" }

 # When Were The Documents Created

> db.richmond.aggregate([{"$project": {"year": {"$substr": ["$created.timestamp", 0, 4]}}}, {"$group": {"_id": "$year", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}])
{ "_id" : "2009", "count" : 289773 }
{ "_id" : "2014", "count" : 112490 }
{ "_id" : "2013", "count" : 101709 }
{ "_id" : "2015", "count" : 22065 }
{ "_id" : "2012", "count" : 16665 }
{ "_id" : "2010", "count" : 9996 }
{ "_id" : "2008", "count" : 5065 }
{ "_id" : "2011", "count" : 4556 }
{ "_id" : "2007", "count" : 36 }
{ "_id" : "2006", "count" : 11 }

The contributions in 2009 is very close to the total contributions from "woodpeck_fixbot", so I ran another query:

> db.richmond.aggregate([{"$match":{"created.user":"woodpeck_fixbot"}}, {$project: {"year":{$substr:["$created.timestamp",0,4]}}}, {$group:{"_id":"$year","count":{$sum:1}}},{$sort:{"count":-1}},{$limit:10}])
{ "_id" : "2009", "count" : 260031 }
{ "_id" : "2010", "count" : 1870 }

 # Top Amenities

> db.richmond.aggregate([{"$match": {"amenity": {"$exists": 1}}}, {"$group": {"_id": "$amenity", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}, {"$limit": 10}])
{ "_id" : "parking", "count" : 730 }
{ "_id" : "restaurant", "count" : 507 }
{ "_id" : "place_of_worship", "count" : 408 }
{ "_id" : "school", "count" : 390 }
{ "_id" : "fast_food", "count" : 266 }
{ "_id" : "fuel", "count" : 166 }
{ "_id" : "bank", "count" : 132 }
{ "_id" : "grave_yard", "count" : 68 }
{ "_id" : "fire_station", "count" : 65 }
{ "_id" : "pharmacy", "count" : 62 }

 # Most Popular Cuisine Among Restaurants

> db.richmond.aggregate([{"$match": {"cuisine": {"$exists": 1}, "$match": {"amenity": "restaurant"}}, {"$group": {"_id": "$cuisine", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}, {"$limit": 5}])
{ "_id" : "chinese", "count" : 14 }
{ "_id" : "mexican", "count" : 14 }
{ "_id" : "italian", "count" : 12 }
{ "_id" : "american", "count" : 10 }
{ "_id" : "pizza", "count" : 9 }
However it's worth noting that only 114 restaurants out of 507 have the field "cuisine", so the accuracy of this query result is quite limited.

 # Most Popular Religion

> db.richmond.aggregate([{"$match": {"amenity": "place_of_worship", "religion": {"$exists": 1}}}, {"$group": {"_id": "$religion", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}, {"$limit": 5}])
{ "_id" : "christian", "count" : 396 }
{ "_id" : "buddhist", "count" : 1 }
{ "_id" : "jewish", "count" : 1 }
{ "_id" : "muslim", "count" : 1 }

 # Most Popular Fast Food Store(By Name)

> db.richmond.aggregate([{"$match": {"amenity": "fast_food"}}, {"$group": {"_id": "$name", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}, {"$limit": 5}])
{ "_id" : "McDonald's", "count" : 28 }
{ "_id" : "Subway", "count" : 20 }
{ "_id" : "Burger King", "count" : 18 }
{ "_id" : "Wendy's", "count" : 14 }
{ "_id" : "Arby's", "count" : 12 }

 # Most Popular Cuisine Among Fast Food

> db.richmond.aggregate([{"$match": {"amenity": "fast_food", "cuisine": {"$exists":

1}}}, {"$group": {"_id": "$cuisine", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}, {"$limit": 5}])
{ "_id" : "burger", "count" : 67 }
{ "_id" : "sandwich", "count" : 28 }
{ "_id" : "chicken", "count" : 21 }
{ "_id" : "pizza", "count" : 17 }
{ "_id" : "mexican", "count" : 7 }

 # Most Popular Banks
> db.richmond.aggregate([{"$match": {"amenity": "bank"}}, {"$group": {"_id": "$name", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}, {"$limit": 5}])
{ "_id" : "Wells Fargo", "count" : 31 }
{ "_id" : "Union First Market Bank", "count" : 15 }
{ "_id" : "Bank of America", "count" : 13 }
{ "_id" : "BB&T", "count" : 12 }
{ "_id" : "SunTrust Bank", "count" : 7 }

Conclusion
I believe the dataset has been cleaned well enough for the purpose of the exercise. A large portion of the dataset was contributed by automated bots. About 50% of the dataset was entered in 2009, which means the dataset is a little dated. An important reason for this is the bot "woodpeck_fixbot" contributed most of its work in 2009, was barely active in 2010, and entirely stopped updating the dataset since then. In fact, 90.86% of the contributions made in 2009 was made by the bot. A little searching online revealed that the bot was run by Frederik Ramm who's a long-time contributor to OSM. He stopped using bot for automatic corrections because he started to believe automated corrections should only be done in exceptional circumstances. Apart from the fixbot, contributions from other users have actually increased significantly over the years.