OpenStreetMap Sample Project Data Wrangling with MongoDB

Wei Bao

Map Area: Cheveland, Ohio, Unit States

https://www.openstreetmap.org/relation/182130#map=11/41.4980/-81.7060
https://mapzen.com/data/metro-extracts/metro/

1. Problems Encountered in the Map

After initially downloading a small sample size of the Cleveland area and I read it by text editor, I noticed four main problems with the data, which I will discuss in the following order:

- problematic characters ("ameni?y")
- Over-abbreviated street names ("N St. Lawrence St.")
- Inconsistent postal codes ("OH44113", "44113-2960")
- "Incorrect" postal codes (Cleveland area zip codes all begin with "441" however a large portion of all documented zip codes were outside this region.)

Problematic characters

I use a regular expression to check the value of attribution in the tags whether there are some problematic characters. For example, "amenity?" actually should be "amenity" and then I can delete problematic character "?" in this character to solve this problem, but there perhaps exist another situation that

“amenity?” become “ameni?y”. How can I fix this trouble? So, I have to see if i have such tags, and if we have any tags with problematic characters, then I can ignore these attributions of tag. I’m sure that it’s not enough good solution, but it’s better to ignore it that keep it.(if you want to fix it in every values that is so difficult and huge workload)

Over-abbreviated Names

The street name and node name always have some abbreviations. I updated all substrings in problematic strings, such that the street name is "N St. Lawrence St.” that should become "North Saint Lawrence Street”. Obviously, the string appear twice “St.”. Firstly I can match last word in the end of string, I change the substring to “Street.” if this word have abbreviation. Secondly, I change other substring to “Saint” because the word “Saint” never put it to the end of string. The other Over-abbreviated words are disposed in a similar way.

Postal Codes

On the other hand, Postal code strings posed a different sort of problem, forcing a decision to strip all leading and trailing characters before and after the main 5-digit zip code. This effectually dropped all leading state characters (as in "OH44113”) and 4-digit zip code extensions following a hyphen ("44113-2960”). I can filter these extra characters by regular expression, no matter what it is. This 5-digit constriction benefits MongoDB aggregation calls on postal codes.

The python code is showed in the following:

```
if re.match(r'^.*([0-9]{5,5}).*$',elem.attrib["v"]):
tmp[elem.attrib["k"].split(":")[0]][elem.attrib["k"].split(":")[1]] =
re.match(r'.*([0-9]{5,5}).*',elem.attrib["v"]).group(1)
```

I just keep 5 digits that be assigned to postcode by regular expression. Regardless, after standardizing inconsistent postal codes, some "incorrect” postal codes surfaced when grouped together with this aggregator.

Sort postcodes by count, descending:

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

Here are the top ten results, beginning with the highest count:

```
[{'_id': '44118', 'count': 21},
 {'_id': '44017', 'count': 19},
 {'_id': '44113', 'count': 14},
```

{'_id': '44094', 'count': 10},

{'_id': '44106', 'count': 9},

{'_id': '44601', 'count': 6},

{'_id': '44074', 'count': 6},

{'_id': '44614', 'count': 5},

{'_id': '44221', 'count': 5},

{'_id': '44109', 'count': 4}]

We need to note that Cleveland area zip codes all begin with “441”, however a large portion of all documented zip codes were outside this region. The above results appear top ten of sorted postcodes by count, but six aren’t even in Cleveland, OH. I am surprised this result and I found that prefix of postal code starting with "440” actually lie in Berea, OH. So, I performed another aggregation to verify a certain suspicion.

Sort cities by count, descending:

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

And, the results are as follows:

[{'_id': 'Cleveland', 'count': 48},

{'_id': 'Cleveland Heights', 'count': 20},

{'_id': 'University Heights', 'count': 11},

{'_id': 'Canton', 'count': 7},

{'_id': 'Cuyahoga Falls', 'count': 7},

{'_id': 'Alliance', 'count': 5},

{'_id': 'Willoughby Hills', 'count': 4},

{'_id': 'North Olmsted', 'count': 3},

{'_id': 'Willoughby', 'count': 3},

{'_id': 'Akron', 'count': 3}]

By the above results, we find the University Heights and Akron that is hometown of NBA superstar LeBron James. These results confirmed my suspicion that this metro extract would perhaps be more aptly named the "Cleveland Metropolitan Area” for its inclusion of surrounding cities in the sprawl. We can expect to acquire the exact data in Cleveland city, rather than big Cleveland area.

So, these postal codes aren’t "incorrect,” but simply unexpected.

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

File name	File sizes
cleveland.osm	206 MB
charlotte.osm.json	238 MB

Number of documents:

```
> db.cleveland.find().count()

1043383
```

Number of nodes:

```
> db. cleveland.find({"type_root":"node"}).count()

971349
```

Number of ways:

```
> db. cleveland.find({"type_root ":"way"}).count()

71522
```

Number of relations:

```
> db. cleveland.find({"type_root ":"relation"}).count()

512
```

Number of unique users:

```
> db.cleveland.distinct("created.user").length

937
```

Top 1 contributing user:

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

[{'_id': 'woodpeck_fixbot', 'count': 586216}]
```

Number of users appearing only once (having 1 post):

```
> db. cleveland.aggregate([{"$group":{"_id":"$created.user",
"count":{"$sum":1}}},
{"$group":{"_id":"$count", "num_users":{"$sum":1}}},
{"$sort":{"_id":1}},
{"$limit":1}])

[{'_id': 1, 'num_users': 206}]
```

"_id" represents postcount

Number of the coffee shop :

```
> db.cleveland.aggregate([{"$match":{"cuisine":{"$exists":1}}},
{"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
{"$match":{"_id":"coffee_shop"}}])

[{'_id': 'coffee_shop', 'count': 14}]
```

Top 10 shop :

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

[{'_id': 'supermarket', 'count': 53},
 {'_id': 'convenience', 'count': 30},
 {'_id': 'car_repair', 'count': 19},
 {'_id': 'doityourself', 'count': 17},
```

{'_id': 'department_store', 'count': 14},

{'_id': 'clothes', 'count': 13},

{'_id': 'hairdresser', 'count': 13},

{'_id': 'bakery', 'count': 8},

{'_id': 'books', 'count': 6},

{'_id': 'butcher', 'count': 6}]

3. Additional Ideas

Data integrality suggestion

I carry out the python code to get out the following results what is the most popular cuisines:

Most popular cuisines:

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

[{'_id': None, 'count': 85},

{'_id': 'american', 'count': 18},

{'_id': 'sandwich', 'count': 12},

{'_id': 'pizza', 'count': 12},

{'_id': 'mexican', 'count': 9},

{'_id': 'italian', 'count': 8},

{'_id': 'burger', 'count': 8},

{'_id': 'ice_cream', 'count': 6},

{'_id': 'diner', 'count': 3},

{'_id': 'seafood', 'count': 3}]

Obviously, the first one in the above result is not correct, the “None” mean that the restaurant may be a having much cuisine’s or not fill the enough data in editing maps information. I don’t expect to get “None” value or NULL, so the contribution of users should as far as possible fill enough data in open street

map forms , then I get out the data to do further data wrangling.

Contributor statistics and gamification suggestion

The contribution of users seems incredibly skewed, possibly due to automate versus manual map editing (the word “bot” appears in some usernames, a “bot” derived from the word “robot”). Here are some user percentage statistics:

- Top user contribution percentage ("woodpeck_fixbot”) – 56.18%
- Combined top 2 users' contribution ("woodpeck_fixbot” and " skorasaurus”) – 60.84%
- Combined Top 10 users contribution – 79.93%
- Combined number of users making up only 1% of posts - 206 (about 21.99% of all users)

Thinking about these user percentages, I’m reminded of "gamification” as a motivating force for contribution. In the context of the OpenStreetMap, if user data were more prominently displayed, perhaps others would take an initiative in submitting more edits to the map. And, if everyone sees that only a handful of power users are creating more than 90% of given map, that might spur the creation of more efficient bots, especially if certain gamification elements were present, such as rewards, badges, or a leaderboard.

Additional data exploration using MongoDB queries

Top 10 appearing amenities:

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

[{'_id': 'place_of_worship', 'count': 1657},
 {'_id': 'school', 'count': 1010},
 {'_id': 'grave_yard', 'count': 509},
 {'_id': 'post_office', 'count': 275},
 {'_id': 'restaurant', 'count': 194},
 {'_id': 'parking', 'count': 128},
 {'_id': 'fast_food', 'count': 124},
 {'_id': 'fuel', 'count': 98},
 {'_id': 'library', 'count': 69},
```

{'_id': 'townhall', 'count': 67}]

Biggest religion:

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

[{'_id': 'christian', 'count': 1609}]
```

Conclusion

After this review of the data, it's obvious that the Cleveland area is incomplete, I just process the small part of this city street information in spite of I believe it has been well cleaned for the purposes of this exercise. It is very appealing to me to a huge scale street data including street, amenity, scenic spot and so on, however this map information have a sort of rough data or invalid data existing. Which one appropriate processor is adopt that is a difficult decision. The purpose of data wrangling is to filter needless data and to keep valuable data. If I have downloaded data is a cleaned data that is perfect result, I don't need to do extra data analyze to extract cleaned data and the result of statistic analysis is such convincing in using this data.