OpenStreetMap Sample Project
Data Wrangling with MongoDB

Jay Teguh Wijaya

Map Area: Bali, Indonesia

http://www.openstreetmap.org/export#map=10/-8.4846/115.1408

http://overpass-api.de/api/map?bbox=114.478,-8.863,115.730,-8.081

Introduction

This article reports my findings from downloading the data from OpenStreetMap.org of Bali area and wrangle with it. The downloaded data are stored in MongoDb database using a small Python script.

The report will start by laying out the problems encountered in the downloaded map, continued with overview of the data, and then some additional data explorations using MongoDb queries.

All scripts used in this report and their outputs are available in appendixes document at the following github repository:
https://github.com/jaycode/ndgree/blob/master/project2/Project2Appendixes.pdf.

1. Problems Encountered in the Map

To audit the data, I did the following steps:

- Initially I was curious about the street names used in the data. Were the naming consistent? Were there incorrect street names? In order to find some leads I wrote a small script audit-streets.py which was a modification to audit.py used in one of the problem sets in the course. Code for audit-streets.py is available in Appendix 1. audit-streets.py.

With this script, I basically printed all street names found in the data before and after being modified by method “update_street_name”.

After running this script, I noticed two main problems with the street names, which I will discuss later in the following order:

- Inconsistent prefixes
- Inconsistent capitalization of first letters (All street names should begin with capital letter).
- Invalid names (“Jl. Bedugul, Sidakarya - Denpasar, Bali - Indonesia”, “Jl. Taman Giri - Banjar Mumbul, Nusa Dua”. Anything after “,” or “-” in should be removed).

Result of this script is available in Appendix 2.

- I thought it would be useful to view a statistics of keys in our dataset. Fortunately there is this tool https://github.com/variety/variety to do just that. In order to use this script, I had to do the following:
    - Importing the data to MongoDB database by running a provisional data.py (this script is available in Appendix 3. data.py).
    - Running variety.js against the newly imported collection with this script:
      ```
      $ mongo test --eval "var collection = 'users', limit = 1"
      variety.js
      ```

  Output of Variety tool can be reviewed in Appendix 4. By running this tool, I found the following insights from our dataset (explanations below):
    - Number of addresses having postal code was very small.
    - Significance of keys.

Inconsistent Prefixes

Unlike street names in the US, street names in Indonesia do not have different suffixes nor prefixes like Road, Avenue, etc. Instead we may either have the same prefix “Jalan” for all road names, or no prefix at all.

Using prefix “Jalan” is preferable to easily separate road names with building names.

HOWEVER, since this is Bali, there is an exception: Some street names do have “Road” or “Street” suffixes, e.g. “Sunset Road” and “Ubud Main Street”. We do not want to add prefix “Jalan” on streets containing these suffixes.

Inconsistent Capitalization

This one is simple. We simply need to capitalize the first letter of any word found in street names.

Invalid Names

Many of the street names mistakenly contained city and area names. We need to remove them. Basically whenever we find special characters after street names, we need to remove any letters after it, for example:

Jl. Bedugul, Sidakarya - Denpasar, Bali - Indonesia => Jalan Bedugul

Small number of addresses with postal codes

Postcode analysis may not be very interesting as we only have 193 rows of them out of 1286 addresses.

Significance of keys

By looking at number of occurrences I was able to find out which keys are statistically significant, which I would then use in my further explorations. For example, since Bali is a tourist resort, we may want to know what attractions are there, and we see that while the key "tourism" has many occurrences, "natural" has also a competing number of occurrences, so we'd better use both.

Value "tree" in key "natural"

An interesting outcome was revealed when I ran a query to find top recorded natural landmarks. I found that the largest number of them (340 nodes) were "tree". Who would've recorded trees in a map, I wondered? So I ran another query to find out which users created them. As I suspected, a whopping 322 of them were created by the same user "okas95". I suspected either a vandalism or bot entry failure has happened here.

To learn about the nature of this problem, I wanted to know how close together were the "tree" nodes inputted by user "okas95". To figure this out, I randomly picked one node and find out its pos, then ran a $near query to it. i.e.

# Number of "tree" nodes within certain area

```
db.openstreet.find({"natural": "tree","created.user": "okas95",
"pos": {"$near": {"$geometry": {"type": "Point", "coordinates":
[115.2102476, -8.6576948]},"$maxDistance": 3800}}}).count()
303
```

The most number of "tree" nodes I could get within the smallest area were 303 nodes within 3.8 kilometers. Expanding the search area ($maxDistance) to 20 kilometers did not add additional nodes. This fact further improved the probability that this data is trash and should not be used in our statistics (as they are clustered together there are higher chance that they were inputted in error and did not show anything useful for us). For the sake of providing an example I did not remove these "tree" nodes from our data, but I will exclude them from our grouping query later.

2. Data Overview

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

File sizes

```
map ......... 50.8 MB
map.json .... 77.7 MB
```

# Number of documents

```
> db.openstreet.find().count()
278378
```

# Number of nodes

```
> db.openstreet.find({"type":"node"}).count()
257440
```

# Number of ways

```
> db.openstreet.find({"type":"way"}).count()
20923
```

# Number of unique users

```
> db.openstreet.distinct("created.user").length
472
```

# Top 1 contributing user

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

[ {"_id" : "Werner17a","count" : 32994} ]
```

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

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

[ {"_id":1,"num_users":78} ] # "_id" represents postcount
```

# Five most contributing users and their percentage of contributions:

```
> db.openstreet.aggregate([{"$group":{"_id":
"$created.user","count":{"$sum": 1}}}, {"$project": { "_id": "$_id",
"count": "$count", "contribution_percentage": {"$divide":
["$count",db.openstreet.count()]} }}, {"$sort": {"count": -1}},
{"$limit":5}])

[ {"_id" : "Werner17a","count" : 32994,"contribution_percentage" :
0.1185222970205979} ...
```

Contributor statistics

The contributions of users seems to be quite skewed as found in the following statistics:

Top user contribution percentage ("Werner17a") - 11.85%
Combined top 5 users' contributions - 40.07%
Combined top 10 users' contributions - 61.14%
Combined number of users making up only 1% (2783) of posts - 306 (64.83% of all users)

Looks like most of the data in this map was provided only by a handful of users. This is perhaps due to the fact that OpenstreetMap project is not very well known in Indonesia. Script to find number of combined number of users making up 1% of all posts is available in Appendix 5. find-combined.py

Additional data exploration using MongoDB queries

# Top 10 appearing amenities:

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

[{"_id" : "restaurant", "count" : 490}, {"_id" : "place_of_worship", "count" : 318}, …

# Biggest religion:

Popular opinion seems to dictate that the biggest religion in Bali is Hindu, followed by Muslim, how is that represented in our openstreetmap data? Let's find out:

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

[{"_id" : "hindu","count" : 241}, {"_id" : "muslim","count" :
21}]
```

# Top 5 natural landmarks
I removed "tree" nodes from the result as explained earlier.

```
> db.openstreet.aggregate([{"$match": {"natural": {"$exists": 1,
"$ne": "tree"}}},
{"$group": {"_id": "$natural", "count": {"$sum": 1}}},
{"$sort": {"count": -1}}, {"$limit": 5}])

[{"_id" : "peak","count" : 276}, {"_id" : "wood","count" : 123},
{"_id" : "coastline","count" : 77}, {"_id" : "beach","count" :
72}, {"_id" : "water","count" : 27}]
```

# Top 5 tourism-related places

```
> db.openstreet.aggregate([{"$match": {"tourism": {"$exists":
1}}}, {"$group": {"_id": "$tourism", "count": {"$sum": 1}}},
{"$sort": {"count": -1}}, {"$limit": 5}])

[{"_id":"hotel","count":457},{"_id":"guest_house","count": 91},
...
```

Conclusion, data validation and tourist map suggestions

After this review of the data I found out that what probably required in the input system of openstreetmap for Bali is a validation mechanism to weed out some entries inputted by error, as demonstrated when we found a large number "tree" nodes in key "natural". This mechanism could perhaps work by a) marking some nodes that are inputted in large number within a relatively nearby area, then b) doing some cleaning based on these marked data. This will require some work to do, but perhaps worth pursuing for cleaner and more useful map data.

On another note, the data we gathered in this project could be used to create a tourist map, maybe in the form of an app or other means, where it would contain a map focusing on providing information relevant for tourists coming to Bali.