OpenStreetMap Project: Data Wrangling with MongoDB
Alf Maglalang

1. Project Overview

The project is to assess the quality of the data from OpenStreetMap, in my project's case, for Bangkok, the capital and most populous city of Thailand with land area of 1900 square kilometers and population of 8,280,925.

Map Area: Bangkok, Thailand. See image bangkok_thailand.png
Name in Thai: กรุงเทพมหานคร in IPA [krūŋ tʰêːp mahǎː nákʰɔ̄ːn]
Map Bounds: minlat=12.661 minlon=99.569 maxlat=15.019 maxlon=101.337

Working File Sizes:
input file: bangkok_thailand.osm: 316.5 Mb
output file: bangkok_thailand.osm.json: 373.8 Mb

Required files:
DataWranglingBangkokOSM.pdf: documentation of wrangling process
bangkok_thailand_sample.osm: 105 Kb, extract from original OSM file
dataprovenance.txt: link to wrangled map and rationale
resources.txt: list of resources to assist with wrangling project
all the lesson 6 exercises: mapparser.py, tags.py, users.py, audit.py, data.py

Code for analyzing and processing OSM file:
gettoptags.py
validatekeys.py
k_attrib_analyzer.py
osm2jsonconverter.py
extractsampleosm.py (code from Udacity's instructor's notes)

Additional files included in project submission:
bkk_pretty_sample.json
invalid_k_values.txt
bangkok_thailand.png
postcode.out
phoneVariations.out

2. Metadata Overview

The input file had the following top-level tags with their corresponding counts. I used the code gettoptags.py to retrieve the following data.

```
{ "node": 1485321,
  "nd": 1754729,
  "bounds": 1,
  "member": 8793,
  "tag": 452337,
  "relation": 913,
  "way": 200230,
  "osm": 1 }
```

For this project, the “node” and “way” elements were analyzed. Since there are 1485321 node tags and 452337 way tags, the JSON file should contain 1685551 documents.

Before converting the OSM file, I examined the key names (“k”) in the “tag” tag element by running validatekeys.py and k_attrib_analyzer.py. The k attribute values fall in the 4 categories.

```
{'alphanum': 414796, 'alphanum_colon': 37244, 'other': 9, 'problemchars': 288}
```

The tagging guidelines from wiki.openstreetmap.org suggest that the tag and attribute elements be alpha-numeric characters using an underscore for more explanatory elements. Compound names can be used using a colon between words like “addr:street”. Of the total k values in the OSM file, 288 had problem characters. I examine the k values more closely by running k_attrib_analyzer.py which created an output of JSON file which I imported into mongoDB.

```
> db.key_names.distinct("key_name").length
977
> db.key_names.find({key_name:/^([a-zA-Z0-9]|_)*(:([a-zA-Z0-9]|_)*)*$/}).count()
941
> db.key_names.find({key_name:{$not:/^([a-zA-Z0-9]|_)*(:([a-zA-Z0-9]|_)*)*$/}}).count()
36
```

977 unique k values represent all the k values in Bangkok's OSM dataset. Off the 977 k values 941 are completely alpha-numeric and underscore or with a colon. And 36 k values have “problem” characters. Sample k values with problem characters are:

```
> db.key_names.find({key_name:{$not:/^([a-zA-Z0-9]|_)*(:([a-zA-Z0-9]|_)*)*$/}}).limit(4)
{ "_id" : "internet_access.source:fee", "total" : 1 }
{ "_id" : "Made by", "total" : 4 }
{ "_id" : "ประเภท", "total" : 1 }
{ "_id" : "name:th-Latn", "total" : 187 }
```

# The top 8 key names are:

```
> db.key_names.find({ }, { "_id" : 0 } ).sort( { "count" : -1 } ).limit(10)
{ "count" : 146234, "key_name" : "highway" }
{ "count" : 54527, "key_name" : "source" }
{ "count" : 35431, "key_name" : "building" }
{ "count" : 33089, "key_name" : "name" }
{ "count" : 17584, "key_name" : "oneway" }
{ "count" : 14236, "key_name" : "name:en" }
{ "count" : 13736, "key_name" : "amenity" }
{ "count" : 9853, "key_name" : "name:th" }
```

Please note that the above is a key_name analysis ONLY. For the actual Bangkok map dataset analysis below, I created a different collection. Before writing to JSON file from the huge Bangkok map OSM dataset, I eliminated all k attribute values that contain problem characters.

3. Data Overview

Using the code osm2jsonconverter.py, I produced a JSON file that will be imported into mongoDB. Please see sample file of documents at bkk_pretty_sample.json
Since both “type” and “address” are being used as values for “k” attribute, I had to use “tagtype” and “bkkaddress” as nodes in the JSON file.

Here are some numbers about the dataset imported into mongoDB.

# Total number of documents
```
> db.bkk2.find().count()
1685551
```

# Number of nodes
```
> db.bkk2.find({"tagtype":"node"}).count()
1485321
```

# Number of ways
```
> db.bkk2.find({"tagtype":"way"}).count()
200230
```

# Number of contributors
```
db.bkk2.distinct( "created.user" ).length
1257
```

# Top 5 contributing users

```
db.bkk2.aggregate([{$group:{"_id":"$created.user", "count":{"$sum":1}}}, {$sort:{"count":-1}},
{$limit:5}])
{ "_id" : "Russ McD", "count" : 264070 }
{ "_id" : "medecember", "count" : 182704 }
{ "_id" : "Paul_012", "count" : 170131 }
{ "_id" : "RocketMan", "count" : 80383 }
{ "_id" : "Parie", "count" : 54815 }
```

# Number of users who made 9 or fewer posts

```
db.bkk2.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}}, {"$group":{"_id":"$count",
"num_users":{"$sum":1}}}, {"$sort":{"_id":1}}, {"$limit":12}])
{ "_id" : 1, "num_users" : 257 }
{ "_id" : 2, "num_users" : 80 }
{ "_id" : 3, "num_users" : 59 }
{ "_id" : 4, "num_users" : 46 }
{ "_id" : 5, "num_users" : 45 }
```

Among the 1257 unique users, 50% of the contributors made an average of ~ 2 posts each. While the top 5 contributors made nearly 45% of the contributions.

4. Problems with Data

# Postal code

According to https://en.wikipedia.org/wiki/Postal_codes_in_Thailand, postal codes in Thailand are 5 digits long. Using grep in bash: `grep -Po "addr:postcode[\" ]*v\=\"(\w*\")" bangkok_thailand.osm > postcode.out`
The output (see postcode.out in supplement files) has some non-5-digit post code. After manipulating data in python code

```
> db.bkk2.find({'bkkaddress.postcode':{$exists:1}}).count()
818
> db.bkk2.find({'bkkaddress.postcode':/^\d{5}$/}).count()
818
```

Of the 818 documents that provide postal code, all now conform to 5-digit. In my code, if all the characters are digits, I grabbed the first 5 digits. If first 5 characters contain non-digit characters, I substituted the entry with the most common Bangkok postcode 10200. This is a simplification that can be changed later with finding the postal code by examining the position coordinates.

# Top 2 postal codes. "10xxx" is the format of Bangkok postal codes.

```
db.bkk2.aggregate([{$group:{"_id":"$bkkaddress.postcode", "count":{"$sum":1}}}, {$sort:{"count":-1}},
{$limit:2}])
{ "_id" : "10200", "count" : 91 }
{ "_id" : "10110", "count" : 83 }
```

# Telephone numbers

To begin the auditing, I used https://en.wikipedia.org/wiki/Telephone_numbers_in_Thailand as a guideline. It was overwhelming to see the different ways phone numbers were entered. (See phoneVariations.out in supplement files.) I also used grep to explore data. In this case the line command I use is: `grep -Po "k\=\"phone\"[ ]*v\=\"(.*\")" bangkok_thailand.osm > phoneVariations.out` . In the sample file, the country code of Thailand is +66 is prefixed many times but incosistently.

I created a new sub-node "countrycode" under node "phone". I also created a sub-node "numbers" that put the subscriber number or numbers in an array. For example, `"+66 38 429450;+66 38 422301"` becomes `{ "phone" : { "numbers" : [ "38 429450", "38 422301" ], "countrycode" : 66 } }`

It did not make sense to separate out the area code. According to Wikipedia, Thailand requires that area codes be dialled whether the number is being called within or outside the metro area.

# Multilingual Issues

Having multilingual data is not a problem itself. But there's a lot of inconsistency in entering data even for “k” attributes. This might cause a problem programmatically. For this project, I retained k attributes that are consistent with OpenStreetMap guidelines of alpha-numeric characters with underscore characters and for compound names with a colon. I do understand that technically Unicode characters should be included as potential “k” attributes. But briefly looking at the Russian and Japanese openstreetmap wiki, it seems that for consistency they are sticking to a “KISS” style guide.

# Street names

Streets in Thai sometimes begin with ถนน (which is usually a main thoroughfare) or ซอย (which branches from one of the main roads). Generally street names are names without any indicator of street type. There is practically no standardized romanized version of Thai streets either in speech or written form. Although used to help non-Thai readers, the romanized names confuse both Thai and non-Thai speakers, especially when the tones are not indicated in romanized names. The messy sample of street names will be a challenging data wrangling task. All street names should be in Thai script. The English or romanized version should be in another field. And in fact, there are other compound fields beginning with “name” and suffixed with a language code, for example, “name_en”, “name_th”, “name_jp”, “name_ru” respectively for English, Thai, Japanese, and Russian

SAMPLE Street Names. The first sample street is an approximate location of the venue including district, sub-district, and province name.

```
{ "bkkaddress" : { "street" : "ทางเลียบถนนกาญจนาภิเษก ต.คลองหนึ่ง อ.คลองหลวง จ.ปทุมธานี " } }
{ "bkkaddress" : { "street" : "Petchkasem Road" } }
{ "bkkaddress" : { "street" : "Siam Cement Road (building 11)" } }
{ "bkkaddress" : { "street" : "Oriental Avenue" } }
{ "bkkaddress" : { "street" : "เพชรเกษม 76/1" } }
{ "bkkaddress" : { "street" : "ถนนเพชรเกษม" } }
{ "bkkaddress" : { "street" : "เพชรเกษม 62/4" } }
{ "bkkaddress" : { "street" : "พุทธมณฑล สาย 1" } }
{ "bkkaddress" : { "street" : "Sukhumvit Soi 16" } }
{ "bkkaddress" : { "street" : "ถนนจรัญสนิทวงศ์" } }
{ "bkkaddress" : { "street" : "New Petchburi Road" } }
{ "bkkaddress" : { "street" : "ถนนพัฒนการ" } }
{ "bkkaddress" : { "street" : "Sukhumvit Road " } }
{ "bkkaddress" : { "street" : "ถนนหลังสวน" } }
{ "bkkaddress" : { "street" : "ถนนคนเดิน สวนจตุจักร โซน2" } }
{ "bkkaddress" : { "street" : "Sukhumvit" } }
```

# Outdated data

There are a lot of outdated data. Bangkok changes rather fast. I have seen the changes – not only with installation of modern transport, but also with creation of many new skyscrapers and businesses. The data should be updated rapidly too.

```
> db.bkk2.find({"created.timestamp":{"$exists":1}}).count()
1685551
> db.bkk2.find({"created.timestamp":{"$gte":"2016-01-01T00:00:00Z"}}).count()
24398
> db.bkk2.find({"created.timestamp":{"$gte":"2015-01-01T00:00:00Z","$lt":"2016-01-
01T00:00:00Z"}}).count()
359307
> db.bkk2.find({"created.timestamp":{"$lt":"2015-01-01T00:00:00Z","$gte":"2014-01-
01T00:00:00Z"}}).count()
367905
```

Only 1.45% of the total 1685551 documents were entered in 2016, 21.32% in 2015. More than 55% of the data were entered in 2013 or earlier.

5. Further Exploration of Dataset

# Top 10 amenities. People who live in Bangkok love to eat. Street food vendors are everywhere. What the Bangkok dataset probably does not include are the thousands of street eating possibilities all over the city. Here are the top 10 amenities in Bangkok.

```
> db.bkk2.aggregate([{$match:{"amenity":{$exists:1}}},{ $group: { "_id": "$amenity", total: { $sum: 1 }
} },{$sort: {"total":-1}} ,{$limit:10}])
{ "_id" : "restaurant", "total" : 2211 }
{ "_id" : "place_of_worship", "total" : 1397 }
{ "_id" : "parking", "total" : 1224 }
{ "_id" : "fuel", "total" : 1004 }
{ "_id" : "bar", "total" : 950 }
{ "_id" : "cafe", "total" : 787 }
{ "_id" : "bank", "total" : 538 }
{ "_id" : "school", "total" : 533 }
{ "_id" : "telephone", "total" : 501 }
{ "_id" : "atm", "total" : 406 }
```

# Top 10 cuisines. 203 cuisines are represented in the dataset

```
> db.bkk2.distinct( "cuisine" ).length
203
> db.bkk2.aggregate([{$match:{"cuisine":{$exists:1}}},{ $group: { "_id": "$cuisine", total: { $sum: 1 }
} },{$sort: {"total":-1}} ,{$limit:10}])
{ "_id" : "thai", "total" : 234 }
{ "_id" : "coffee_shop", "total" : 127 }
{ "_id" : "japanese", "total" : 85 }
{ "_id" : "burger", "total" : 81 }
{ "_id" : "pizza", "total" : 53 }
{ "_id" : "indian", "total" : 50 }
{ "_id" : "chicken", "total" : 48 }
{ "_id" : "italian", "total" : 46 }
{ "_id" : "regional", "total" : 41 }
{ "_id" : "international", "total" : 37 }
```

# 9 religions are represented in the dataset. Nearly 95% of Thais are Buddhist. This explains why the Buddhist representation in the dataset is around 90%.

```
> db.bkk2.distinct( "religion" ).length
9
> db.bkk2.aggregate([{$match:{"religion":{$exists:1}}},{ $group: { "_id": "$religion", total: { $sum: 1
} } },{$sort: {"total":-1}}])
{ "_id" : "buddhist", "total" : 1191 }
{ "_id" : "christian", "total" : 53 }
{ "_id" : "muslim", "total" : 45 }
{ "_id" : "taoist", "total" : 13 }
{ "_id" : "hindu", "total" : 13 }
{ "_id" : "buddist", "total" : 2 }
{ "_id" : "sikh", "total" : 2 }
{ "_id" : "spiritualist", "total" : 1 }
{ "_id" : "jewish", "total" : 1 }
```

6. Ways of Improving and Analyzing the Data

Analyzing big data is a problem when memory resources are limited. I had to use the bash shell in my Linux to get around this problem. In addition to the simple pager methods like “more” or “less” suggested by the Udacity coach, I also used the powerful and fast “grep” command with regular expressions. What I could not do at all in any text editors in OSX or Windows, it took seconds in my small Linux box. For this project, I primarily used regular expressions in Linux to audit both the OSM dataset and JSON output file. The use of the regular expressions in my python code for processing is a copy of regular expressions used in “grep”.

Upon analyzing the data, the biggest problem I saw are formatting issues and multilingual issues. The formatting of postal codes or phone numbers for example can be fixed by implementing validation code as data is being entered. The multilingual problem is primarily due to inconsistency in data entry – some in Thai and some in Romanized version. One of the most prolific contributor as indicated above, Paul_012, inconsistently uses both Thai and romanized version.

```
> db.bkk2.find({'bkkaddress.street':{$exists:1},'created.user':'Paul_012'},
{'_id':0,'bkkaddress.street':1,'created.user':1})
{ "created" : { "user" : "Paul_012" }, "bkkaddress" : { "street" : "ถนนสีลม" } }
{ "created" : { "user" : "Paul_012" }, "bkkaddress" : { "street" : "Silom Road" } }
{ "created" : { "user" : "Paul_012" }, "bkkaddress" : { "street" : "Ngamwongwan 44" } }
{ "created" : { "user" : "Paul_012" }, "bkkaddress" : { "street" : "Rajadamnern Nok Avenue" } }
{ "created" : { "user" : "Paul_012" }, "bkkaddress" : { "street" : "เจริญกรุง 40" } }
```

In fact, the top 2 samples **ถนนสีลม and Silom Road** are actually the same road. I am wondering if this contributor is literate in both languages. I am assuming that he or she is not.

```
> db.bkk2.find({'name':{$exists:1}}).count()
32715
> db.bkk2.find({'name':/น/}).count()
11790
```

A majority of the nodes are in English or Roman alphabet. Of the 32715 documents that have “name”s, only about a third uses Thai script. This brings up the question who exactly is the audience for this openstreetmap in Bangkok. Who are the contributors? Who is setting the standards other than openstreetmap?

Before gamification or incentivization can be discussed, there has to be some sort of collaboration between literate Thai speakers and non-Thai speakers. Guidelines must be set. This linguistic confusion cannot be encouraged. I have not looked at the Russia and Japan datasets. But the fact that Russia and Japan have wiki.openstreetmap.org in Russian and Japanese respectively is an indicator that guidelines exist. The contributors for the Thai dataset should follow suit.