OpenStreetMap Sample Project

Data Wrangling with MongoDB

Paulo Luis Franchini Casaretto

Map Area: Vale do Itajaí and Florianópolis Metropolitan Area, Santa Catarina, Brazil

http://overpass-api.de/api/map?bbox=-49.4385,-27.8488,-48.3508,-26.8094

Downloading the data

I've included a script to download the data automatically. It's named `download-data`. It downloads the chosen area and saves it to data/raw-data.osm.

After downloading the data, I use the script called `importtomongo` to import the whole json into a mongo collection using the format suggested in the Case Study for the course. Here is an example for a node representing a city.

(Interesting fact: this city was recently the stage for a Volvo Ocean Race stopover)


```
{
  "name": "Itajaí",
  "created": {
    "uid": "591237",
    "changeset": "29912562",
    "version": "13",
    "user": "daltux",
    "timestamp": "2015-04-01T19:02:35Z"
  },
  "is_in:country": "Brazil",
  "is_in:country_code": "BR",
  "pos": [
    -26.9117879,
    -48.6669852
  ],
  "rank": "30",
  "place": "city",
  "is_in:continent": "South America",
  "population": "172081",
  "type": "node",
  "id": "273316",
  "is_in": "Santa Catarina, Brazil"
}
```


Data overview

File sizes

- data/raw-data.osm 76 MB

- data/raw-data.osm.json 84 MB

Characteristics

Number of documents

`db.udacity.find().count()` 384064

Number of nodes

`db.udacity.find({"type":"node"}).count()` 349776

Number of ways

`db.udacity.find({"type":"way"}).count()` 34268

Number of unique users

`db.udacity.distinct({"created.user"}).length` 410

Top 3 contributing users

```
db.udacity.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$sort":{"count": -1 }}, {"$limit":3}])
{ "_id" : "Prblanco", "count" : 83015 }
{ "_id" : "Rachmaninoff", "count" : 57591 }
{ "_id" : "DrSeuthberg", "count" : 31098 }
```

Number of users appearing only once (having 1 post)

```
db.udacity.aggregate([
  {"$group":{"_id":"$created.user", "count":{"$sum":1}}},
  {"$group":{"_id":"$count", "num_users":{"$sum":1}}},
  {"$sort":{"_id":1}},
  {"$limit":1}
])
{ "_id" : 1, "num_users" : 65 }
```

Auditing

After getting a feeling for the data looking at the raw osm data, I used an iterative approach creating and refining scripts to audit and fix each piece of the data.

Street names

Using a modified version of the script developed on https://www.udacity.com/course/viewer#!/c-ud032-nd/l-768058569/e-865319708/m-900198650 I investigated problems with street names.

Problems found

At first I included only a few street "types" and got this as output.

```
{u'Alameda': set([u'Alameda Adolfo Konder',
                  u'Alameda C\xe9sar Nascimento',
                  u'Alameda Duque de Caxias']),
 u'Almirante': set([u'Almirante Barroso', u'Almirante Tamandar\xe9']),
 u'Av.': set([u'Av. Coleira',
              u'Av. Governador Ivo Silveira',
              u'Av. Hilza Terezinha Pagani',
              u'Av. Max de Souza',
              u'Av. Teporti',
              u'Av. Thiago Antunes Teixeira']),
 u'Avendia': set([u'Avendia Jorge Lacerda']),
 u'Beco': set([u'Beco Argentina']),
 u'Dom': set([u'Dom Jaime Camara']),
 u'Escadaria': set([u'Escadaria da Rua Pedro Soares']),
 u'Estrada': set([u'Estrada Intendente Ant\xf4nio Damasco',
                  u'Estrada Jornalista Jaime de Arruda Ramos',
                  u'Estrada Roz\xe1lia Paulina Ferreira']),
 u'Koesa': set([u'Koesa']),
 u'Lauro': set([u'Lauro M\xfcller']),
 u'Namy': set([u'Namy Deeke']),
 u'Pra\xe7a': set([u'Pra\xe7a Pereira Oliveira', u'Pra\xe7a XV de Novembro']),
 u'Quarta': set([u'Quarta Avenida']),
 u'Rodovia': set([u'Rodovia "Seu Chico" Francisco Thomaz dos Santos',
                  u'Rodovia Admar Gonzaga',
                  u'Rodovia Amaro Ant\xf4nio Vieira',
                  u'Rodovia Ant\xf4nio Luiz Moura Gonzaga',
                  u'Rodovia BR 101 esq. c/Rua Maria Helena Kretzer',
                  u'Rodovia BR-101 - Km 210 ( pr\xf3x. SESI/SENAC)',
                  u'Rodovia BR-101, Km 116,',
                  u'Rodovia BR-101, Km 116,8',
                  u'Rodovia BR-101, Km 156,3 - s/n\xb0',
                  u'Rodovia BR-101, Km 164',
                  u'Rodovia BR-101, Km 195',
                  u'Rodovia BR-101, Km 198, n\xba 1.173',
                  u'Rodovia BR-101, Km 210 - s/n\xba',
                  u'Rodovia Baldicero Filomeno',
                  u'Rodovia Gilson da Costa Xavier',
                  u'Rodovia Governador M\xe1rio Covas',
                  u'Rodovia Haroldo Soares Glavan',
                  u'Rodovia Jos\xe9 Carlos Daux',
                  u'Rodovia Jos\xe9 Carlos Daux (SC-401)',
                  u'Rodovia Jo\xe3o Gualberto Soares',
                  u'Rodovia SC 401, km 4',
                  u'Rodovia SC-401',
```

```
                 u'Rodovia SC-401, km 5',
                 u'Rodovia SC-405',
                 u'Rodovia Virg\xedlio V\xe1rzea']),
 u'SC': set([u'SC 401', u'SC 401, Km 01']),
 u'Salom\xe9': set([u'Salom\xe9 Damazio Jacques']),
 u'Santos': set([u'Santos Saraiva']),
 u'Serv.': set([u'Serv. Jos\xe9 Cardoso de Oliveira']),
 u'Servid\xe3o': set([u'Servid\xe3o Ab\xedlio Silva',
                      u'Servid\xe3o Ant\xf4nio Arlindo Bitencourt',
                      u'Servid\xe3o Expedicion\xe1rio Napole\xe3o',
                      u'Servid\xe3o Jaborandi',
                      u'Servid\xe3o Luiz Pedro Pereira',
                      u'Servid\xe3o Maria Cordeiro Fernandes',
                      u'Servid\xe3o Placido Ferreira',
                      u'Servid\xe3o da Prainha',
                      u'Servid\xe3o das \xc1guias']),
 u'S\xe3o': set([u'S\xe3o Miguel']),
 u'Terceira': set([u'Terceira Avenida']),
 u'Travessa': set([u'Travessa Ratclif',
                   u'Travessa da Liberdade',
                   u'Travessa dos Imigrantes']),
 u'Via': set([u'Via Expressa no Mercado BIG'])}
```

I fed the right types back into the script and ran it again.

```
{u'Almirante': set([u'Almirante Barroso', u'Almirante Tamandar\xe9']),
 u'Av.': set([u'Av. Coleira',
              u'Av. Governador Ivo Silveira',
              u'Av. Hilza Terezinha Pagani',
              u'Av. Max de Souza',
              u'Av. Teporti',
              u'Av. Thiago Antunes Teixeira']),
 u'Avendia': set([u'Avendia Jorge Lacerda']),
 u'Dom': set([u'Dom Jaime Camara']),
 u'Escadaria': set([u'Escadaria da Rua Pedro Soares']),
 u'Koesa': set([u'Koesa']),
 u'Lauro': set([u'Lauro M\xfcller']),
 u'Namy': set([u'Namy Deeke']),
 u'Quarta': set([u'Quarta Avenida']),
 u'SC': set([u'SC 401', u'SC 401, Km 01']),
 u'Salom\xe9': set([u'Salom\xe9 Damazio Jacques']),
 u'Santos': set([u'Santos Saraiva']),
 u'Serv.': set([u'Serv. Jos\xe9 Cardoso de Oliveira']),
 u'S\xe3o': set([u'S\xe3o Miguel']),
 u'Terceira': set([u'Terceira Avenida']),
 u'Via': set([u'Via Expressa no Mercado BIG'])}
```

The one major problem with the data was abbreviated street types. To fix that, I had to replace:

- Av -> Avenida
- Serv. -> Servidão

The minor issues found were:

Postfixed 'Avenida'

In my hometown there are two avenues that when translated literally mean Third Avenue, and Fourth Avenue. People colloquially refer to them in this way (postfixing 'Avenida'). I checked other tools (Google and Apple Maps) to see how they represented them and chose to change them to prefixing. 'Avenida'

Missing street types

There were some street that had no type information. I used other tools to check each one individually.

The final code for the audit and fix can be found at (respectively):

- `audit/street_names`
- `fix/street_names`

Postal codes

The second step was to analyze postcodes. I decided this time to not use python and use mongo shell client directly. The file `audit/postal-codes` outputs 4 values.

- Number of properly formated postal-codes (e.g. 88338-220)
- Number of records with the most common alternate form (e.g 88338220)
- Number of records that do not fit either of the alternatives
- Number of records that do not start with 88 or 89 (expected for the given area)

Problems found

As expected there were some (52) records with the alternative form. I decided to change all of them to the proper format using `fix/postal-codes`.

There were no records that did not fit either formats and only one that did not start with the expected prefix. This record was fixed manually.

Other problems

I noticed there were some nodes that had information about speed cameras but the data that held the max speed for the camera was saved as a string. I used the following snippet to fix this.

```
db.udacity.find({"highway":"speed_camera"}).forEach(function(doc) {
    doc.maxspeed = new NumberInt(doc.maxspeed);
    db.udacity.save(doc);
});
```

Improvement Ideas

The dataset has one major problem. There just isn't enough data. The number of documents is very low for such a big area that includes the capital for the state.

Despite the efforts of some hard working users, the mapping community for the area is still very weak and much of the cities (specially outside the capital) have nothing but road information.

One interesting project to fix this situation would be to study how the mapping communities evolved in the areas that have well developed maps and apply the same techniques for the chosen area.

To achieve that goal, one could explore the listing of territory based projects and get in touch with project leaders and learn from them what kind of activities would help jumpstart the local community.

More specifically, the Brazilian directory might have useful info on how the current mapping effort is going and what would be the best way to help.

Additional data exploration using MongoDB queries

Top 10 appearing amenities

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

Biggest religion

```
db.udacity.aggregate([
  {"$match":{"amenity":{"$exists":1}, "amenity":"place_of_worship"}},
  {"$group":{"_id":"$religion", "count":{"$sum":1}}},
  {"$sort":{"count": -1}},
  {"$limit":1}
])
[ {"_id":"christian","count":155} ]
```

Most popular cuisines

```
db.udacity.aggregate([
  {"$match":{"amenity":{"$exists":1}, "amenity":"restaurant"}},
  {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
  {"$sort":{"count":-1}},
  {"$limit":3}
```

```
])
{ "_id" : null, "count" : 166 }
{ "_id" : "pizza", "count" : 36 }
{ "_id" : "seafood", "count" : 25 }
```

Here we can see the information for restaurants is very poor.

Average maxspeed for speed cameras

```
db.udacity.aggregate([
  {"$match":{"highway":"speed_camera"}},
  {"$group":{"_id":null, "average_maxspeed":{"$avg": "$maxspeed"}}},
])
{ "_id" : null, "average_maxspeed" : 54 }
```

Conclusion

In spite of the deficiencies pointed earlier, the dataset was successfully cleaned and could be used as a standard for future contributors.