Project: Wrangle OpenStreetMap Data

Data Wrangling with MongoDB Jocelyn Moreau

Map Area: Finistere sud, Bretagne, France

https://export.hotosm.org/en/v3/exports/382e36c5-7823-4848-b664-6fcdc6b533ee (https://export.hotosm.org/en/v3/exports/382e36c5-7823-4848-b664-6fcdc6b533ee)

The OSM is 50,1Mo.

In this project, we are going to address the following points:

1. Prepare and sample the OSM file
   - Number of tags (mapparser.py)
   - Analyse the problematic tags (tags.py)
   - Audit the file (audit.py)
2. Import in MongoDB and requested analysis
3. Additional Statistics and Ideas
   - Contributor statistics
   - Additional data exploration using MongoDB
4. Conclusion

1. Prepare and sample the OSM file

For this project, I decided to work on a region of France, called Finistere sud, in Bretagne, France, a nice area for vacation! The full map is available as france-finistere-sud_export.osm (Size of 50,1Mo, so compliant with the >50Mo size) in the current Github repository. I extracted a sample using the sample_file.py script (see cell below). The result is saved in france-finistere-sud_export_sample.osm. This is the file I will use for the next steps.

1.1 Number of tags (mapparser.py)

Counting the number of nodes for each file (using mapparser.py script):

Full osm file: {'member': 49, 'meta': 1, 'nd': 275503, 'node': 216346, 'note': 1, 'osm': 1, 'relation': 24, 'tag': 74529, 'way': 34610}

Sample file: {'member': 49, 'meta': 1, 'nd': 275503, 'node': 216346, 'note': 1, 'osm': 1, 'relation': 24, 'tag': 74529, 'way': 34610}

I then analysed the following 4 points using the script2-tags.py:

Legend:

- "lower", for tags that contain only lowercase letters and are valid,
- "lower_colon", for otherwise valid tags with a colon in their names,
- "problemchars", for tags with problematic characters, and
- "other", for other tags that do not fall into the other three categories.

Result of the scripts for the full OSM file: {'lower': 74018, 'lower_colon': 463, 'other': 42, 'problemchars': 6}

This first analysis shows us that there is:

- 74018 tags having no upper case, which means that will have to add an uppercase for the first letter of the tag in the next step.
- 463 tags with a colon (a short look shows these are web addresses)
- 6 problematic characters in the map chosen
- 42 others, which seem to be fine (they are tags for mentioning the source of the information, SIREN (French unique reference number for company) ref. ...

  ### 1.3 Audit the file (audit.py) After auditing the sample file with the audit.py script, I could not detect any error... so I decided to run the audit on the complete OSM file.

The data has a very good quality, very few mistake were detected.

Lower case mispelling:

- rue -> Rue

I updated the audit.py dictionnary to include the detected mistake and ran the script again to correct all these mispelling.

Abbreviations: There was no abbreviation in the map chosen (either a student from Udacity applied this training to correct the mistake, or Openstreetmap implemented a script to correct it) --> nothing to do here.

I could not find any mistake in the followings tags: city name, building names

Phones: all phone numbers don'thave the same format, some with spaces, others without. I updated the script to perform a correction of the phone numbers (remove spaces and parenthesis).

2. Import in MongoDB

2.1 Transform the xml into json format ()

I used the following 4-data_json.py script that turns the xml into json, cleaning the mistakes detected using the previously used functions.

2.2 Import the generated json into MongoDB

I imported the json file (70,1 Mo) size into a MongoDB database (called francefinisteresud), using the mongoimport library. As a GUI for the database requests, I will use MongoBooster and copy paste below the result of the requests.

2.3 Data overview

Let's check the data we imported:

Number of documents

```
db.francefinisteresud.find({}).count()
```

-> 250950

Number of unique users

```
db.francefinisteresud.distinct('created.user').length
```

-> 90 differents users

Number of nodes

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

-> 216346 results (which corresponds to the number of nodes we counted in the xml file)

Number of ways

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

-> 34604 (instead of 34610: 6 ways were not imported, error message was as following: key ref:clochers.org must not contain '.', I will not try to fix this error for now)

Number of schools:

```
db.francefinisteresud.find({'others.amenity': "school"}).count() --> 10
```

Number of restaurants

```
db.francefinisteresud.find({'others.amenity': "restaurant"}).count() --> 21
```

Additional Statistics and Ideas

Top 10 users

> db.francefinisteresud.aggregate([{' group': {'_id': ' created.user', 'count': {' sum' : 1}}}, {' sort': {'count' : -1}}, {'$limit': 10}])

The top ten users is as follow:

id	count
eric_G	82709
Michelwald	54536
isnogoud	49702
osmmaker	21470
Super-Map	11550
PierenBot	10877
jo2929	8962
luiswoo	2358
lcroc	2165
Fifi Fofo	1518

count of each different amenity referenced:

> db.francefinisteresud.aggregate([{' group': {'_id': ' others.amenity', 'count': {' sum' : 1}}}, {' sort': {'count' : -1}}])

-> here is the result: (the first result with null is the total number of nodes without tag amenities)

_id	count
null	250854
place_of_worship	23
restaurant	21
school	10
pharmacy	6
bank	4
toilets	5
townhall	3

_id	count
fuel	3
cafe	3
fast_food	3
bar	2
fire_station	2
public_building	2
car_wash	1
parking	1
community_centre	1
post_office	1
cinema	1
marketplace	1
pub	1
police	1
recycling	1

Additional ideas

As the xml format is flexible we could think of adding many more information, linked to the different amenities listed in the map. For restaurant, for example, users could add reviews of the food, pictures, comments (the way google is doing it).

1. Benefits: the same database could be used by many apps to show on a map different informations. This would bring more users to visit openstreetmap and use the platform.
2. Anticipated problems: of courses, this would imply a good moderator system, not to publish incorrect data and make sure that the comments are fair for example, that the picture are the one showing the restaurant... and on top, the size of the database (one extract of an area is generally over 1 Go) would go through the roof, implying more cost for openstreetmap.

Another idea could be to implement a layer, on top of the map, that shows with different colors the level of completeness of the map, helping willing users to work on the porrly documented areas first. For the area I analysed with coordinate between -3,99272 and -3,72681 (middle point being -3,834) and 47,7684 and 47,94464 (middle point 47,85), I could divide the area in 4 squares and count per squares the number of nodes.

1. Benefits: it would make it easy for users willing to help to give priority on areas with less data first, and ensure that the maps are in average well documented
2. Anticipated problems: some areas (country side) don't have a lot to show compared to cities where a lot of information is available. So the color system would not really represent the degree of completeness of each areas.

Zone 1 (lower left area)

```
db.francefinisteresud.find({ and: [{'pos.0': { lt: 47.85}, {'pos.1': {$lte: -3.834}}]}).count() -> 38449
```

Zone 2 (upper right area)

```
db.francefinisteresud.find({ and: [{'pos.0': { gte: 47.85}, {'pos.1': {$gt: -3.834}}]}).count() ->
33351
```

Zone 3 (lower right area)

```
db.francefinisteresud.find({ and: [{'pos.0': { lt: 47.85}, {'pos.1': {$gt: -3.834}}]}).count() -> 22044
```

Zone 4 (lower right area)

```
db.francefinisteresud.find({ and: [{'pos.0': { gte: 47.85}, {'pos.1': {$lte: -3.834}}]}).count() ->
122502
```

The total matches the total number of nodes of 216346. Here we can see that even on a small area like this one, the number of nodes per area is huge.

Conclusion

The area audited, cleaned and analysed, showed a very good level of data integrity. Very few mistakes were found. One easy improvment on openstreetmap side, would be to automatically tranform the first letter of a street name to an upper case. This project allowed me to learn a new database language (MongoDB) which is completly different from SQL, but allowing to store data in a much more flexible way, although it can become very messy, very quickly, if no standards are followed.

References

- All python scripts used to audit, clean and analyse the data come from the case study provided in the previous lesson
- the mongoDB queries were built using https://docs.mongodb.com (https://docs.mongodb.com/)
- queries were executed using Mongoboost software