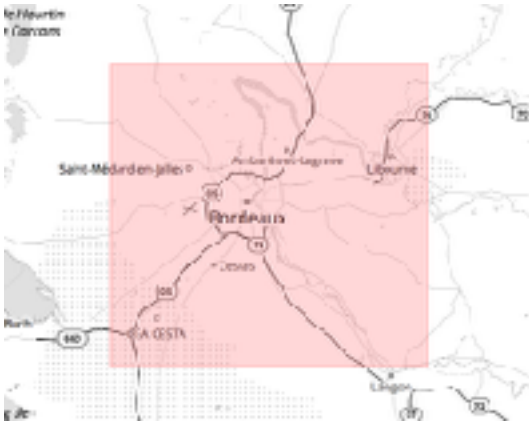
P3 Data Wrangling: Wrangle OpenStreetMap Data MongoDB - Bordeaux (France)

by Quentin THOMAS

Introduction



In this third project, I use the data munging techniques studied during the courses to clean OpenStreetMap data for Bordeaux, city of France.

Dataset: https://mapzen.com/data/metro-extracts/metro/bordeaux_france/

https://en.wikipedia.org/wiki/Bordeaux

Even if I live in Paris, I have chosen Bordeaux in order to have a smaller set of data which allow me to work easily on my personal computer. In addition I will love to live one day in this beautiful city.

Audit

Sample file

To get a first idea of the kind of cleaning required and test my first functions, I have used the sample.py script with a k equal to 40. Then I have increased k step by step.

Python script used

In order to clean the data and create the JSON file for importing the data into a MongoDB database, I have used the two files studied during the course:

- audit.py : For auditing the data and creating the mapping dictionary
- data.py : For creating the JSON file
- cleaning.py: For cleaning and fixing the data
- process_map.py: For processing the xml file

Due to some french particularities I had to modify both script and created some new functions.

Problems encountered

Street names

Street names were not homogeneous. Different cases have been appeared:
- Case issues (such as AVENUE or avenue, or route for Route)
- Incomplete street Name (such as Imp for Impasse)
- Abbreviations issue (such as Ctre for Center)

Code sample

```
Street_expected = [u"All\xe9e", "Avenue",
"Chemin", u"Cit\xe9", "Clos", "Cours",
"Esplanade", "Impasse", "Place",
        "Route", "Rue", "Boulevard", "Lieu",
"Quai","BIS","Bassin","Centre","Lieu-dit"]

mapping = { "Imp": "Impasse",
        "rue": "Rue",
        "quai": "Quai",
        "place": "Place",
        "route": "Route",
        "cours": "Cours",
        "boulevard" : "Boulevard",
        "lieu": "Lieu",
        "AVENUE": "Avenue",
        "avenue": "Avenue",
        "Allee": u"All\xe9e",
        u"all\xe9e":u"All\xe9e",
        u'all\xe9es':u"All\xe9e",
        u"All\xe9es":u"All\xe9e",
        "Av": "Avenue",
        "C.Cial" : "Centre Commercial",
        "Ctre" : "Centre",
        "Lieu-Dit":"Lieu-dit",
        "lieu-dit":"Lieu-dit",
        "esplanade":"Esplanade"
        }
```

Phone numbers

Phone numbers were fully heterogeneous. It was possible to find many shapes:
- 05 XX XX XX XX
- 05XXXXXXXX
- 5XXXXXXXX
- 33 5 XX XX XX XX
- +33XXXXXXXXX
- etc …

Most of the time, the right way to present european phone numbers is with the international code first and space between number:

+33 (0)5 XX XX XX XX

I have created a function to manage all the phone number which do not match with this format.

Code sample

```
def update_phone_number(phone):

# I need at least 9 number at the end of the string

   if (re.findall(r'[1-9]([ .-]?[0-9]{2}){4}$', phone)):
      phone = phone.replace(" ","")
      i = len(phone)
            clean_phone  ="+33 (0)"+phone[i-9]+" "+phone[i-8:i-6]+" "+phone[i-6:i-4]+"
"+phone[i-4:i-2]+" "+phone[i-2:i]
   else:
      clean_phone = ""

   return clean_phone
```

Overview of the data

File sizes

bordeaux_france.osm ..... 1.14 GB
bordeaux_france.osm.json .... 1.62 GB

Database size

```
> show databases;

admin        0.000GB
dbBordeaux   0.437GB
local        0.000GB
```

Number of unique users

```
> db.bdx.distinct('created.user').length

1553
```

Best contributor

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

{ "_id" : "mides", "count" : 697615 }
```

Number of documents

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

```
5528257
```

Number of nodes

```
> db.bdx.find({‘type':'node'}).count()
```

```
4708592
```

Number of ways

```
> db.bdx.find({‘type':'way'}).count()
```

```
816460
```

Top 5 Banks


```
> db.bdx.aggregate([{$match:{'amenity':{$exists:1},
'amenity':'bank'}},{$group:{'_id':'$name' ,’count':{$sum:
1}}},{$sort:{'count':-1}},{$limit:5}])
```

```
{ "_id" : "Crédit Mutuel", "count" : 37 }
{ "_id" : null, "count" : 28 }
{ "_id" : "Crédit Agricole", "count" : 27 }
{ "_id" : "Caisse d'Épargne", "count" : 25 }
{ "_id" : "BNP Paribas", "count" : 19 }
```

Other ideas about the dataset

Food service in Bordeaux

I am wondering about what it is possible to eat in bordeaux.

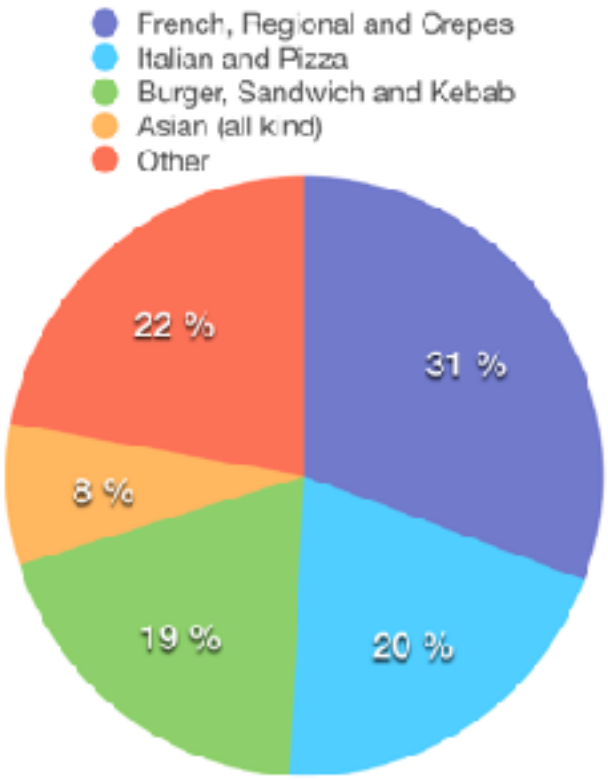
```
> db.bdx.find({‘cuisine':{$exists:1}}).count()
```

```
604
```

So there are 604 place for eating with a type of cuisine defined in the database. Let’s see the top 20 of the cuisine offer.

```
> db.bdx.aggregate([{$match:{'cuisine':{$exists:1}}},{$group:
{'_id':'$cuisine', 'count':{$sum:1}}},{$sort:{'count':-1}},
{$limit:20}])
```

```
{ "_id" : "french", "count" : 112 }
{ "_id" : "pizza", "count" : 84 }
{ "_id" : "burger", "count" : 67 }
{ "_id" : "regional", "count" : 66 }
{ "_id" : "italian", "count" : 38 }
{ "_id" : "kebab", "count" : 31 }
{ "_id" : "sandwich", "count" : 19 }
{ "_id" : "japanese", "count" : 16 }
{ "_id" : "asian", "count" : 15 }
{ "_id" : "chinese", "count" : 14 }
{ "_id" : "international", "count" :
8 }
{ "_id" : "crepe", "count" : 7 }
{ "_id" : "vietnamese", "count" : 7 }
{ "_id" : "american", "count" : 6 }
{ "_id" : "spanish", "count" : 6 }
{ "_id" : "sushi", "count" : 5 }
{ "_id" : "chicken", "count" : 5 }
{ "_id" : "seafood", "count" : 4 }
{ "_id" : "turkish", "count" : 4 }
{ "_id" : "fish_and_chips", "count" :
4 }
```



I am very surprised by the number of burger, sandwich and kebab cuisine. By looking deeper into the database I can see that there are two main kind of restauration place.

```
> db.bdx.find({'amenity':'restaurant'}).count()

709

> db.bdx.find({'amenity':'fast_food'}).count()

179
```

By talking about fast food, I want to have an overview and to know the company with the most restaurants.

```
> db.bdx.aggregate([{$match:{'amenity':{$exists:1},
'amenity':'fast_food'}},{$group:{'_id':'$name' ,'count':{$sum:
1}}},{$sort:{'count':-1}},{$limit:5}])
```

```
{ "_id" : "McDonald's", "count" : 26 }
{ "_id" : null, "count" : 21 }
{ "_id" : "Quick", "count" : 9 }
{ "_id" : "KFC", "count" : 6 }
```

```
{ "_id" : "Subway", "count" : 3 }
```

No big surprise here, Macdonald is historically well implemented in France. I also notice that there are 21 fast food not identified. Is it the same thing for the restaurants?

```
> db.bdx.aggregate([{$match:{'amenity':{$exists:1}, 'name':
{$exists:0}, 'amenity':'restaurant'}},{$group:
{'_id':'$amenity' ,'count':{$sum:1}}},{$sort:{'count':-1}},
{$limit:20}])

{ "_id" : "restaurant", "count" : 57 }
```

As for the fast food restaurants, there are a lot of restaurants not identified.
It is a pity to have a localized point of interest but not its name. Let's see if it concerns a lot of amenity.

```
db.bdx.aggregate([{$match:{'amenity':{$exists:1}, 'name':{$exists:
0}}},{$group:{'_id':'$amenity' ,'count':{$sum:1}}},{$sort:
{'count':-1}},{$limit:20}])

{ "_id" : "parking", "count" :
1839 }
{ "_id" : "bench", "count" :
944 }
{ "_id" : "recycling",
"count" : 566 }
{ "_id" : "waste_basket",
"count" : 549 }
{ "_id" : "post_box", "count" :
454 }
{ "_id" : "bicycle_parking",
"count" : 445 }
{ "_id" : "atm", "count" :
159 }
{ "_id" : "drinking_water",
"count" : 128 }
{ "_id" : "vending_machine",
"count" : 125 }
{ "_id" : "toilets", "count" :
124 }
{ "_id" : "place_of_worship",
"count" : 88 }
{ "_id" : "fuel", "count" :
83 }
{ "_id" : "parking_entrance",
"count" : 76 }
{ "_id" : "townhall", "count" :
66 }
{ "_id" : "shelter", "count" :
66 }
{ "_id" : "grave_yard", "count"
: 60 }
{ "_id" : "restaurant", "count"
: 57 }
{ "_id" : "pharmacy", "count" :
57 }
{ "_id" : "parking_space",
"count" : 51 }
{ "_id" : "fountain", "count" :
47 }
```

In function of the amenity, the presence or absence of a name is more or less important.

For parking or toilets that kind of information is completely useless but for a restaurant, a bank it could be essential.

For adding a point, a user has to register as an editor on the Openstreetmap website and then fill a form for each data he want to add.

For some kind of data it could be interesting to have mandatory fields to fill before accepting a new record.

The main problem with this solution is that it could discourage some user for this non profit work. However for some type of POI, specificities are priceless.

Maybe some prefilled list with courant names like Mac Donalds or kind of cuisines like Pizza or Burger can help to have more homogeneous data and push the editor to add more details.

Files included

- report.pdf
- audit.py
- data.py
- cleaning.py
- process_map.py
- sample.py
- bdx_sample.osm

Bibliography

N/A