Data Wrangling with MongoDB

Final Project

Pavle Guduric

Contents

Introduction

In this project I analyzed data available at the OpenStreetMap. I have chosen the area of *Frankfurt am Main*, Germany since I live and work there. *Frankfurt* is the fifth largest city in Germany and the financial center of the European Union. The goal of the project was to clean and prepare the data so I can import it into MongoDB and use it to analyze different city districts.

Problems Encountered in the Map

The region I have chosen resulted in a 207 MB file. This does not necessarily depend on the size of the region but on the metadata available for that region. So if you chose a densely populated map you should expect a larger OSM file [1]. To take a first look at the data you will need an editor that can handle large XML files, such as Notepad++. Also make sure you have enough RAM installed and free when you execute your Python scripts.

I want to analyze the city of Frankfurt and its districts only. Since I didn't precisely define the city boundaries when I exported the data from the OpenStreetMap, I have to remove the neighboring cities. The following query reveals that these cities are indeed included in my data set:

> **db.cities.distinct("address.city")**

```
[u'Frankfurt',                u'Mühlheim',                  u'Mühlheim/Main',
 u'Offenbach am Main',        u'Steinbach (Taunus)',        u'Mühlheim am Main',
 u'Frankfurt am Main',        u'Frankfurt-Hausen',          u'Frankfurt a. M.',
 u'Eschborn',                 u'Frankfurt-Ostend',          u'Schwalbach am Taunus',
 u'Frankfurt/M',              u'Frankfurt am Main -         u'Schwalbach',
 u'Frankfurt am Main -Nied',  Griesheim',                   u'Rumpenheim',
 u'Bad Vilbel',               u'Maintal-Bischofsheim',      u'Offenbach - Waldheim',
 u'Offenbach',                u'Offenbach-Rumoenheim',      u'Frankfurt-Rödelheim']
 u'Maintal',                  u'Rödelheim',
 u'Frankfurt/Main',           u'Offenbach-Bürgel',
```

The cities that are not part of the Frankfurt are: *Eschborn*, *Offenbach am Main*, *Bad Vilbel*, *Maintal*, *Steinbach*, *Schwalbach* and *Rumpenheim*.

Another problem is that multiple names are being used for *Frankfurt* such as *Frankfurt*, *Frankfurt am Main*, *Frankfurt/M* and *Frankfurt/Main*. I would like to have a single name *Frankfurt am Main*, so I need to map other names to this one. City districts listed above such as *downtown*, *Nied*, *Hausen*, *Ostend*, *Griesheim* and *Rödelheim* also need to have their names corrected and prefixed with the *Frankfurt am Main* instead of *Frankfurt* only.

Postal codes of the cities I excluded need to be removed as well. There are total 47 postal codes according to the following query:

> db.cities.distinct("address.postcode")

```
"60322",   "60594",   "60488",
"60431",   "60486",   "60528",
"60320",   "60389",   "60388",
"60435",   "63477",   "63069",
"60439",   "60596",   "60313",
"60433",   "60327",   "60487",
```

"60318",
"60325",
"65936",
"65934",
"60599",
"60314",
"60329",
"60316",
"65929",
"60385",
"60529",
"60489",
"60326",
"65760"
"63067",
"60386",
"60323",
"60598",
"60311",
"63065",
"63075",
"65933",
"63073",
"60306",
"60308",
"61449",
"65824",
"63071",
"60310"

All postal codes that are not assigned to *Frankfurt am Main* or one of its districts (marked bold) are removed from the data set. Also, city (district) names for some of the nodes are not precise enough. For example there are post codes where city (district) name is only *Frankfurt am Main* as shown by the following query:

```
> db.cities.aggregate({"$group":{"_id":"$address.city",
                                 "postcodes":{"$addToSet":"$address.postcode"}}})
{ "_id" : "Frankfurt am Main - Rödelheim", "postcodes" : [ "60489" ] }
{ "_id" : "Frankfurt am Main - Ostend", "postcodes" : [ "60314" ] }
{ "_id" : "Frankfurt am Main - Hausen", "postcodes" : [ "60487" ] }
{ "_id" : "Frankfurt am Main - Nied", "postcodes" : [ "65934" ] }
{ "_id" : "Frankfurt am Main", "postcodes" : [ "60308", "60598", "60311", "60323", "60433", "60529",
"60326", "60386", "60385", "60329", "60320", "60489", "60431", "65934", "60596", "60594",
"60318", "60487", "60327", "60316", "60306", "60388", "65936", "60486", "60528", "60439",
"60314", "60389", "60488", "60325", "60322", "60435", "60310", "60313", "60599" ] }
{ "_id" : null, "postcodes" : [ "60487", "60431", "60326", "60435", "60313", "60327", "60599",
"60325", "60322", "60329", "60489", "65934", "60596", "60318", "60316", "60486", "60528",
"60311", "60323", "60389", "60314", "60385", "60320", "60594", "60488", "65936", "60433",
"60439" ] }
```

There are also nodes that have post code but no city (district) name. In order to fix this, I have to create a mapping between post codes and districts. By doing so I can assign correct city (district) to each node where it is necessary.

Majority of the phone numbers have international format. I corrected those that do not so that each number starts with German international code +49. I also removed special signs like ()/- and empty spaces so that each phone string contains only numbers with prefix +.

As data cleaning is iterative process, I will be importing temporary data sets into MongoDB before I import the final one. I am doing so because I can query it more easily.

Final Data Set

Final data set has 31 districts and one generic place holder (*Frankfurt am Main*) for nodes without post code:

```
> db.cities.distinct("address.city")
[
```

```
        "Frankfurt am Main - Dornbusch",
        "Frankfurt am Main - Ginnheim",
        "Frankfurt am Main - Westend-Nord",
        "Frankfurt am Main - Berkersheim",
        "Frankfurt am Main - Heddernheim",
        "Frankfurt am Main - Frankfurter Berg",
        "Frankfurt am Main - Sachsenhausen-Nord",
        "Frankfurt am Main - Rödelheim",
        "Frankfurt am Main - Seckbach",
        "Frankfurt am Main - Gutleutviertel",
        "Frankfurt am Main - Praunheim",
        "Frankfurt am Main - Niederrad",
        "Frankfurt am Main",
        "Frankfurt am Main - Bergen-Enkheim",
        "Frankfurt am Main - Altstadt",
        "Frankfurt am Main - Hausen",
        "Frankfurt am Main - Nordend-West",
        "Frankfurt am Main - Bockenheim",
        "Frankfurt am Main - Sossenheim",
        "Frankfurt am Main - Nied",
        "Frankfurt am Main - Oberrad",
        "Frankfurt am Main - Fechenheim",
        "Frankfurt am Main - Bahnhofsviertel",
        "Frankfurt am Main - Nordend-Ost",
        "Frankfurt am Main - Bornheim",
        "Frankfurt am Main - Schwanheim",
        "Frankfurt am Main - Griesheim",
        "Frankfurt am Main - Riederwald",
        "Frankfurt am Main - Westend-Süd",
        "Frankfurt am Main - Sachsenhausen-Süd",
        "Frankfurt am Main - Opernturm",
        "Frankfurt am Main - Innenstadt"
]
```

The query below shows districts with corresponding post codes:

```
> db.cities.aggregate({"$group":{"_id":"$address.city", "postcodes":{"$addToSet"
:"$address.postcode"}}})
{ "_id" : "Frankfurt am Main - Innenstadt", "postcodes" : [ "60310", "60308" ] }
{ "_id" : "Frankfurt am Main - Opernturm", "postcodes" : [ "60306" ] }
{ "_id" : "Frankfurt am Main - Westend-Süd", "postcodes" : [ "60323" ] }
{ "_id" : "Frankfurt am Main - Riederwald", "postcodes" : [ "60386" ] }
{ "_id" : "Frankfurt am Main - Griesheim", "postcodes" : [ "60326" ] }
{ "_id" : "Frankfurt am Main - Nordend-Ost", "postcodes" : [ "60316" ] }
{ "_id" : "Frankfurt am Main - Rödelheim", "postcodes" : [ "60489", "60486" ] }
{ "_id" : "Frankfurt am Main - Bergen-Enkheim", "postcodes" : [ "60388" ] }
{ "_id" : "Frankfurt am Main - Bahnhofsviertel", "postcodes" : [ "60329" ] }
{ "_id" : "Frankfurt am Main - Niederrad", "postcodes" : [ "60528" ] }
{ "_id" : "Frankfurt am Main - Hausen", "postcodes" : [ "60487" ] }
{ "_id" : "Frankfurt am Main - Fechenheim", "postcodes" : [ "60314" ] }
{ "_id" : "Frankfurt am Main - Bornheim", "postcodes" : [ "60385" ] }
{ "_id" : "Frankfurt am Main - Sossenheim", "postcodes" : [ "65936" ] }
```

{ "_id" : "Frankfurt am Main - Schwanheim", "postcodes" : ["60529"] }
{ "_id" : "Frankfurt am Main - Bockenheim", "postcodes" : ["60325"] }
{ "_id" : "Frankfurt am Main - Nordend-West", "postcodes" : ["60318"] }
{ "_id" : "Frankfurt am Main - Berkersheim", "postcodes" : ["60435"] }
{ "_id" : "Frankfurt am Main - Ginnheim", "postcodes" : ["60431"] }
{ "_id" : "Frankfurt am Main - Westend-Nord", "postcodes" : ["60320"] }
{ "_id" : "Frankfurt am Main - Sachsenhausen-Nord", "postcodes" : ["60596", "60594"] }
{ "_id" : "Frankfurt am Main - Heddernheim", "postcodes" : ["60439"] }
{ "_id" : "Frankfurt am Main - Gutleutviertel", "postcodes" : ["60327"] }
{ "_id" : "Frankfurt am Main - Oberrad", "postcodes" : ["60599"] }
{ "_id" : "Frankfurt am Main - Nied", "postcodes" : ["65934"] }
{ "_id" : "Frankfurt am Main - Altstadt", "postcodes" : ["60311", "60313"] }
{ "_id" : "Frankfurt am Main", "postcodes" : [60311] }
{ "_id" : "Frankfurt am Main - Frankfurter Berg", "postcodes" : ["60433"] }
{ "_id" : "Frankfurt am Main - Sachsenhausen-Süd", "postcodes" : ["60598"] }
{ "_id" : "Frankfurt am Main - Dornbusch", "postcodes" : ["60322"] }
{ "_id" : "Frankfurt am Main - Seckbach", "postcodes" : ["60389"] }
{ "_id" : "Frankfurt am Main - Praunheim", "postcodes" : ["60488"] }

Note that some larger districts have more than one post code. Now that each post code has been assigned to its city district, I have enough data to do some analysis per districts.

Overview of the Data

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

File Size

- frankfurt.osm - 207 MB
- frankfurt.osm.json - 214 MB

> **db.stats()**

- before cleaning

{
"db" : "ffm",
"collections" : 3,
"objects" : 996964,
"avgObjSize" : 299.55469605722976,
"dataSize" : 298645248,
"storageSize" : 335917056,
"numExtents" : 16,
"indexes" : 1,
"indexSize" : 32360608,
"fileSize" : 469762048,
"nsSizeMB" : 16,
"dataFileVersion" : {
"major" : 4,
"minor" : 5
},
"extentFreeList" : {

```
            "num" : 0,
            "totalSize" : 0
        },
        "ok" : 1
    }
```


- after cleaning


```
    {
        "db" : "ffm",
        "collections" : 3,
        "objects" : 981410,
        "avgObjSize" : 295.5836357893235,
        "dataSize" : 290088736,
        "storageSize" : 335917056,
        "numExtents" : 16,
        "indexes" : 1,
        "indexSize" : 31853696,
        "fileSize" : 469762048,
        "nsSizeMB" : 16,
        "dataFileVersion" : {
            "major" : 4,
            "minor" : 5
        },
        "extentFreeList" : {
            "num" : 0,
            "totalSize" : 0
        },
        "ok" : 1
    }
```


Different Types of Tags

This is the result produced by the *count_tags* method available in the C*ountTags.py* file:


```
{
        'bounds': 1,
         'member': 120523,
         'meta': 1,
         'nd': 1166251,
         'node': 830666,
         'note': 1,
         'osm': 1,
         'relation': 3201,
         'tag': 732513,
         'way': 166294
}
```


Note that there is no before/after comparison as the analysis was done based on the input *Frankfurt.osm* xml file.

Number of Documents

> db.cities.count()

- before cleaning: 996960
- after cleaning: 981406

Number of Nodes

> **db.cities.find({“type”:”node”}).count()**

- before cleaning: 830590
- after cleaning: 829054

Number of Ways

> **db.cities.find({“type”:”way”}).count()**

- before cleaning: 166272
- after cleaning: 152266

Number of Unique Users

> **db.cities.distinct("created.user").length**

- before cleaning: 1298
- after cleaning: 1279

Top 1 Contributing User

> **db.cities.aggregate(**
{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$sort":{"count":-1}},
{"$limit":1})

- before cleaning: { "_id" : "MichaH", "count" : 423002 }
- after cleaning: { "_id" : "MichaH", "count" : 414970 }

Number of users appearing only once (having 1 post)

> **db.cities.aggregate(**
{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
{"$group":{"_id":"$count", "num_users":{"$sum":1}}},
{"$sort":{"_id":1}},
{"$limit":1})

- before cleaning: {"_id":1,"num_users":328}
- after cleaning: {"_id":1,"num_users":324}

Other Ideas about the Dataset

Frankfurt is very popular migration destination. Every year it gets richer for about 10.000 new residents [2]. One of the most popular online services for relocation and apartment searching is immobilienscout24. You can look for apartments by using different criteria such as street or district, diameter of the preferred area, size, renting cost... Also you are limited to the information about the apartment and its location that is provided in an ad. You have to research about the location by using different online maps or web sources like Wikipedia.

We could use the information available in the OpenStreetMap to improve the searching experience. We could search for an apartment by exploring different characteristics of a city or a district. For example, if we like to go out often, we would look for a location with lot of restaurants or cafes; if we have children we would look for a place with schools nearby. We would not be limited to the information provided in the ad but to the collective opinion of the majority. An algorithm could propose us a location based on our preferences instead of having to perform manual research of the location.

This section shows some of the statistics that could be used as the apartment search metric.

Top Amenity

The top reported amenity is parking. *Frankfurt* is the fifth largest city in Germany and is the center of the *Rhein-Main* region. It has approx. 700.000 citizens, with commuters slightly over 1 million [3]. *Frankfurter Kreuz* is the most heavily used interchange in the European Union [4].

```
> db.cities.aggregate([{"$match":{"amenity":{"$exists":1}}},
        {"$group": {"_id": {"Amenity":"$amenity"}, "count": {"$sum": 1}}},
        {"$project": {'_id':0, "Amenity":"$_id.Amenity", "Count":"$count"}},
        {"$sort": {"Count": -1}},
        {"$limit" : 5 }])
```

{u'Amenity': u'parking', u'Count': 1070},
{u'Amenity': u'restaurant', u'Count': 888},
{u'Amenity': u'bicycle_parking', u'Count': 506},
{u'Amenity': u'bench', u'Count': 496},
{u'Amenity': u'recycling', u'Count': 367}

Top Dining Districts

The most visited districts by tourists are *Altstadt* [5] and *Bornheim* [6] – or merry village as it is called, so no wonder those districts have the most restaurants and cafes in Frankfurt.

```
> db.cities.aggregate([{"$match":{"amenity":{"$exists":1}, "address.city":{"$exists":1},
                                "amenity":{"$in":["cafe","restaurant"]}}},
        {"$group":{"_id":{"District":"$address.city"}, "Total": {"$sum":1}}},
        {"$project":{"_id":0, "District":"$_id.District", "Count":"$Total"}},
        {"$sort":{"Count":-1}},
        {"$limit":3}])
```

{u'Count': 28, u'District': u'Frankfurt am Main - Altstadt'},
{u'Count': 23, u'District': u'Frankfurt am Main'},
{u'Count': 23, u'District': u'Frankfurt am Main - Bornheim'}

Top Cuisine

As we can see it is a dead race between *Indian* and *Italian*. I can confirm that from my rich experience ☺

```
> db.cities.aggregate([{"$match":{"amenity":{"$exists":1}, "address.city":{"$exists":1},
                                "cuisine":{"$exists":1}, "amenity":"restaurant",
                                "address.city":district}},
        {"$group":{"_id":{"District":"$address.city", "Food":"$cuisine"}, "Total": {"$sum":1}}},
        {"$project":{"_id":0, "District":"$_id.District", "Food":"$_id.Food", "Count":"$Total"}},
        {"$sort":{"Count":-1}},
        {"$limit":2}])
```

{u'Count': 2, u'District': u'Frankfurt am Main - Altstadt', u'Food': u'indian'},
{u'Count': 2, u'District': u'Frankfurt am Main - Altstadt', u'Food': u'regional'}

{u'Count': 5, u'District': u'Frankfurt am Main - Bornheim', u'Food': u'regional'},
{u'Count': 3, u'District': u'Frankfurt am Main - Bornheim', u'Food': u'italian'}]

Family Friendly Districts

If you have kids then you should consider living in *Rödelheim*, *Sachsenhausen-Nord* or *Seckbach* as they have more schools than the other districts.

```
> db.cities.aggregate([{"$match":{"amenity":{"$exists":1}, "address.city":{"$exists":1},
                                  "amenity":{"$in":["school"]}}},
        {"$group":{"_id":{"District":"$address.city", "Type":"$amenity"}, "Total": {"$sum":1}}},
        {"$project":{"_id":0, "District":"$_id.District", "Count":"$Total"}},
        {"$sort":{"Count":-1}},
        {"$limit":3}])
```

{u'Count': 9, u'District': u'Frankfurt am Main - Rödelheim' },
{u'Count': 9, u'District': u'Frankfurt am Main - Sachsenhausen-Nord' },
{u'Count': 8, u'District': u'Frankfurt am Main - Seckbach' }

Top Bakeries

There are many bakeries in *Frankfurt* and below are the top 3 that cover most of the city.

```
> db.cities.aggregate( [{"$match":{"shop":{"$exists":1}, "shop":"bakery", "name":{"$exists":1}}},
                        {"$group":{"_id":{"District":"$address.city", "Name":{"$toLower": "$name"}},
"count":{"$sum":1}}},
                        {"$project": {'_id':0, "District":"$_id.District", "Name":"$_id.Name",
"Count":"$count"}},
                        {"$sort":{"Count":-1}}, {"$limit":3}])
```

{u'Count': 13, u'Name': u'eifler'},
{u'Count': 10, u'Name': u'glocken bäckerei'},
{u'Count': 10, u'Name': u'wiener feinbäcker'}

Conclusion

Cleaning and preparing data is cumbersome and sometimes boring task. A good set of tools is necessary. I used Excel, Notepad++, Vim, python and MongoDB. You can manage data set complexity by doing it iteratively, processing successively small portions of your data set that you understand well. Iterative process makes sense since most of data schemas are extended at some point in time.

Given the population of the city (~700.000), I would say that 1279 (0.18%) unique users is not very much. I was also expecting to see more amenities like restaurants and cafes in *Altstadt* and *Bornheim.* According to www.bornheim-frankfurt.de there are more than just 23 restaurants and cafes in the *Bergerstraße*(*Bornheim*) alone. I was also surprised to see very few *Indian* and *Italian* restaurants in the *Altstadt* and *Bornheim.* This leads me to a conclusion that *Frankfurt* area is not complete.

Works Cited

[1] "OSM," [Online]. Available: http://wiki.openstreetmap.org/wiki/OSM_XML.

[2] [Online]. Available: http://www.frankfurt.de/sixcms/detail.php?id=2811&_ffmpar[_id_inhalt]=7524.

[3] "FFMI," [Online]. Available: http://frankfurt-interaktiv.de/frankfurt/geschichte/fakten.html.

[4] "WikiFFM," [Online]. Available: http://en.wikipedia.org/wiki/Frankfurt.

[5] "WikiFFMA," [Online]. Available: http://en.wikipedia.org/wiki/Altstadt_%28Frankfurt_am_Main%29.

[6] "WikiFFMB," [Online]. Available: http://en.wikipedia.org/wiki/Bornheim_%28Frankfurt_am_Main%29.