OpenStreetMap Sample Project
Data Wrangling with MongoDB
Marcel van der Poel

Map Area: Geldermalsen and surroundings, Netherlands

1. Problems Encountered in the Map

Because I am familiar with dutch geography and the dutch metro area maps from Map Zen are really large, I choose a custom map from my own municipality of Geldermalsen and surroundings. In the python code I created a number of lists and sets to help me understand the data and some routines at the end of the code to print them out.

Postal Codes, Street and City

In the Netherlands every postal code refers to a street and a city. A (large) street can consists of multiple postal codes. I wanted to check if the postal codes in my dataset exist and if the corresponding addresses and cities were correct. The strategy I choose was to use the official source for postal code information (www.postcode.nl). I found out that using their API is not for free, therefore I wrote some Python code (ScrapePostalCodeInformation.py which I included) to scrape the site. I added all postalcode to a set (unique postal codes) , then used the set to retrieve the data and stored the result in a file Postalcode.csv. Example first three rows:

```
PostCode|Street|City
4153AM|Voorstraat|Beesd
4116GR|Hoge Kornseweg|Buren
4116GE|Graafschapsstraat|Buren
```

- I found out that some postal codes had the structure '9999 XX' instead of '9999XX'. This is solved in the Wrangle.py program.
- I also found out that postal code 4191CG that has only one address in OSM and that points to a boat, is non existent on the postal code site. I chose to add this one manually to the csv-file, since on a number of internet sites the shipping company uses it as a postal address.

- All cities and their spelling were correct, but some streets were spelled differently, using abbreviations on the postal code website (e.g. Dr. instead of Doctor). I adjusted the csv file to not use the abbreviations.
- Another thing was that the postalcode website uses latin-1 encoding and OSM uses utf-8 encoding. I tried various strategies but was unable to compare the two programmatically (spent a lot of hours on it) and finally decided to filter out the postal codes, since the data was correct.
- Finally, 218 addresses had a wrong postal code vs street/house number/city combination. In a number of cases the same postal code was used for two different pairs of street/city combinations in the OSM-data. This is not allowed. Another observation was that it seems some street/city combinations were 'flipped', meaning giving one combination the postal code of the other. Finally, in case of street 'Zeelt' with postal code 4191CP, this is a construction site with no houses yet build.
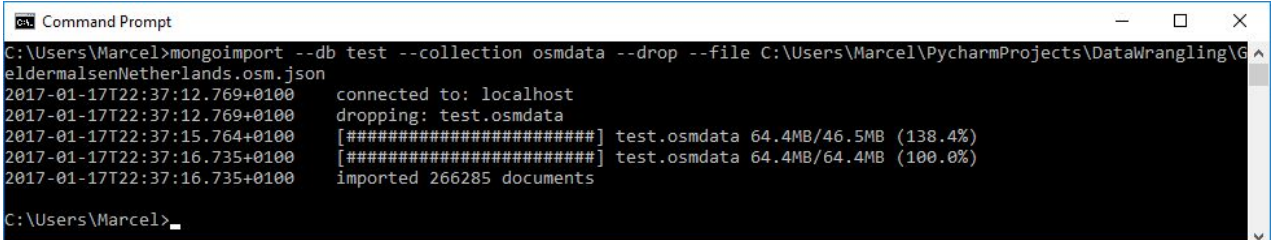
Multiple columns

I used the columns in the XML tags to group a number of tags as a separate object in the JSON file, because higher level. I found there were a number of tags where the number of columns was three were I felt the extra level did not add any additional information. I removed the one of the three levels by removing the mid level of by connecting the second and third level.

Descriptions in multiple languages

There were a few tags that next to a notation in a standard language, also new notations in multiple other languages. Not all languages were used every time, mostly none of them were used and I did not find them useful for any analysis. Therefore, I removed them from the JSON file.

I imported the JSON file using: mongoimport --db test --collection osmdata --drop --file C:\Users\Marcel\PycharmProjects\DataWrangling\GeldermalsenNetherlands.osm.json. I

```
Command Prompt                                                    —   □   ×

C:\Users\Marcel>mongoimport --db test --collection osmdata --drop --file C:\Users\Marcel\PycharmProjects\DataWrangling\G
eldermalsenNetherlands.osm.json
2017-01-17T22:37:12.769+0100    connected to: localhost
2017-01-17T22:37:12.769+0100    dropping: test.osmdata
2017-01-17T22:37:15.764+0100    [#######################] test.osmdata 64.4MB/46.5MB (138.4%)
2017-01-17T22:37:16.735+0100    [#######################] test.osmdata 64.4MB/64.4MB (100.0%)
2017-01-17T22:37:16.735+0100    imported 266285 documents

C:\Users\Marcel>_
```

Churches

I made the following list of all the building types

```
Command Prompt - mongo                                              —  □  ✕
> db.osmdata.aggregate( [ {$match: {'building': {$exists:true}}}, {$group: {_id:'$building',
count:{$sum:1}}} ] )
{ "_id" : "windmill", "count" : 1 }
{ "_id" : "office", "count" : 5 }
{ "_id" : "yes;apartments", "count" : 1 }
{ "_id" : "apartments", "count" : 199 }
{ "_id" : "houseboat", "count" : 3 }
{ "_id" : "bakery (industrial)", "count" : 1 }
{ "_id" : "yes;house", "count" : 4 }
{ "_id" : "house", "count" : 10030 }
{ "_id" : "commercial", "count" : 134 }
{ "_id" : "garage", "count" : 25 }
{ "_id" : "industrial", "count" : 279 }
{ "_id" : "apartments;commercial", "count" : 1 }
{ "_id" : "retail", "count" : 100 }
{ "_id" : "church", "count" : 1 }
{ "_id" : "static_caravan", "count" : 39 }
{ "_id" : "city gate", "count" : 1 }
{ "_id" : "school", "count" : 3 }
{ "_id" : "greenhouse", "count" : 8 }
{ "_id" : "public", "count" : 1 }
{ "_id" : "construction", "count" : 100 }
Type "it" for more
>
```

I saw that the number of churches and schools was far too low, so I created a text index on the whole dataset (db.osmdata.createIndex({ "$**": "text" })) and search for the text 'church'. I still got one result, but saw in that result the fields 'denomination' and 'religion'. I also saw in the osm, that in my area of investigation there were multiple crosses where there are also churches. Denomination seems to be the field that identifies a church:

```
Select Command Prompt - mongo                                      —  □  ✕
> db.osmdata.aggregate( [ {$match: {'denomination': {$exists:true}}}, {$group: {_id:'$denomin
ation', count:{$sum:1}}} ] )
{ "_id" : "roman_catholic", "count" : 3 }
{ "_id" : "Hervormde Gemeente", "count" : 1 }
{ "_id" : "protestant", "count" : 10 }
>
```

Religion returned 4 hits for Jewish graveyards and 16 churches. In that search I saw that 'amenity' can have the value 'place_of_worship'.

```
Command Prompt - mongo                                             —  □  ✕
> db.osmdata.aggregate( [ {$match: {'amenity': 'place_of_worship'}}, {$group: {_id:'$amenity'
, count:{$sum:1}}} ] )
{ "_id" : "place_of_worship", "count" : 16 }
>
```

My conclusion from this is that deriving conclusions from the data is not so straightforward as it might seem; multiple fields seem to be a candidate to determine if a building is a church. After some investigation amenity seems to be the best candidate.

I had the same experience with restaurants. There I first looked at the field 'cuisine'. But I found the number of results didn't match my own experience of the neighbourhood. Further investigation showed that here 'amenity'='restaurant' is a better choice. What did 'disappoint' me was the fact that it almost all cases the 'cuisine' was missing, it was clear what the cuisine was, e.g. a restaurant named "Greek restaurant Odyssey II" had no mentioning of the cuisine.

2. Data Overview

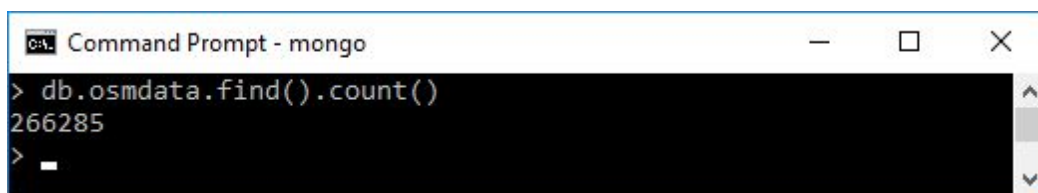This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

File sizes

GeldermalsenNetherlands.osm: 54,6 MB (57.347.821 bytes)
GeldermalsenNetherlands.osm.json: 64,4 MB (67.539.948 bytes)
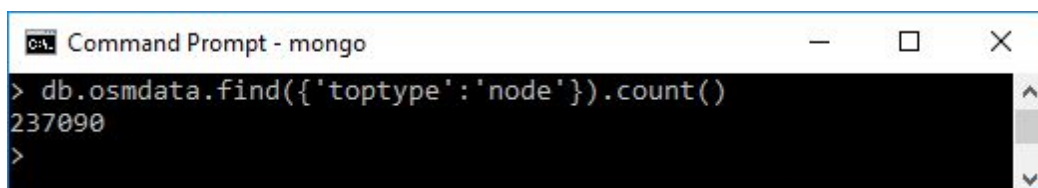
Number of wrong postal codes
218 (see python code)

Number of documents
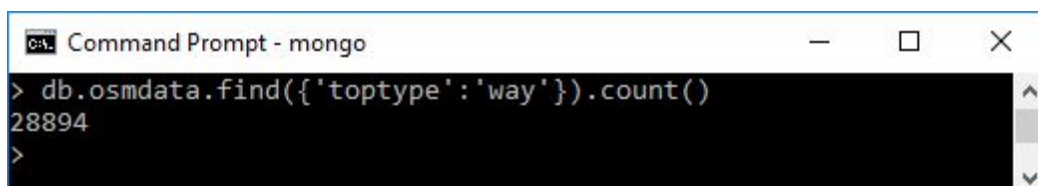
```
Command Prompt - mongo                          —    □    ×
> db.osmdata.find().count()
266285
>
```

Number of nodes

```
Command Prompt - mongo                          —    □    ×
> db.osmdata.find({'toptype':'node'}).count()
237090
>
```

Number of ways

```
Command Prompt - mongo                          —    □    ×
> db.osmdata.find({'toptype':'way'}).count()
28894
>
```

Number of relations

```
Command Prompt - mongo                               —   □   ×
> db.osmdata.find({'toptype':'relation'}).count()
301
>
```

Number of unique users

```
Command Prompt - mongo                               —   □   ×
> db.osmdata.distinct('created.user').length
137
>
```

Number of mutations on osmdata per year

```
Command Prompt - mongo                                          —   □   ×
> db.osmdata.aggregate( [ {$project: {'year': { $substr: [ '$created.timestamp', 0, 4 ] }}} ,
{$group: {_id:'$year', count:{$sum:1}}},{$sort:{_id:1}}] ).pretty()
{ "_id" : "2007", "count" : 234 }
{ "_id" : "2008", "count" : 97 }
{ "_id" : "2009", "count" : 4097 }
{ "_id" : "2010", "count" : 59178 }
{ "_id" : "2011", "count" : 2080 }
{ "_id" : "2012", "count" : 1979 }
{ "_id" : "2013", "count" : 2503 }
{ "_id" : "2014", "count" : 171993 }
{ "_id" : "2015", "count" : 864 }
{ "_id" : "2016", "count" : 23260 }
>
```
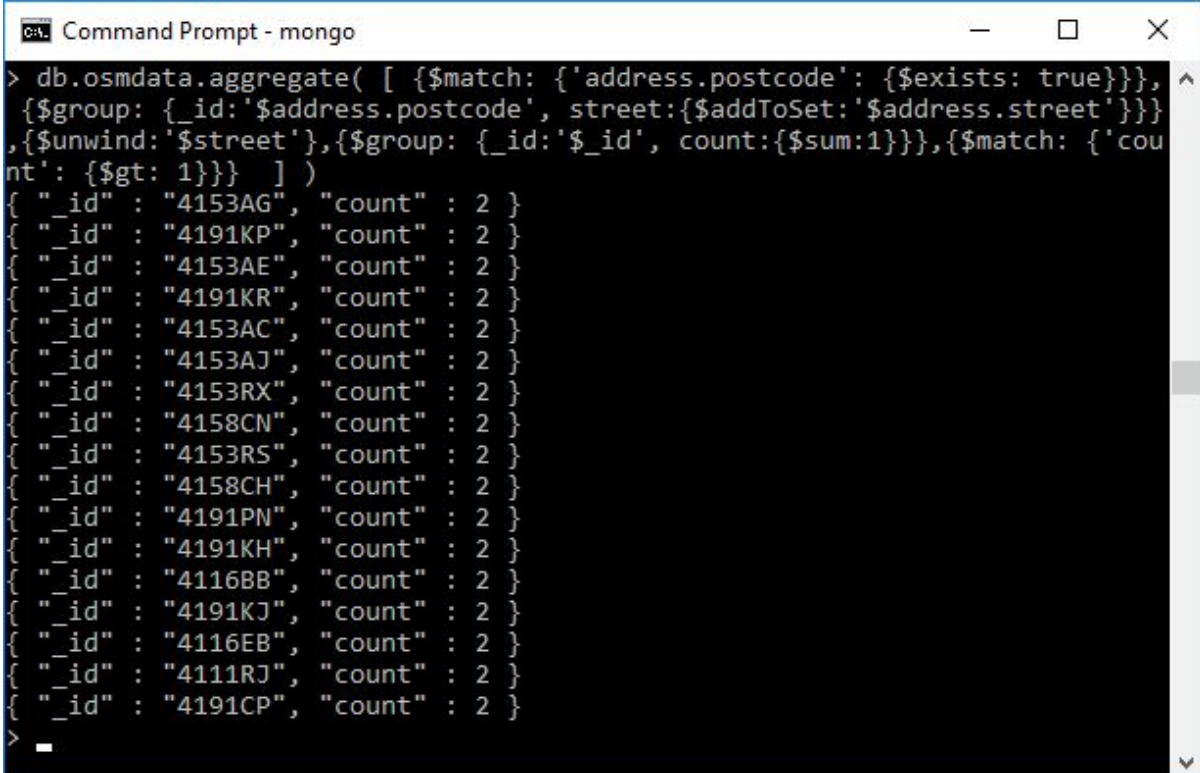
Year with the highest number of mutations on osmdata

```
Command Prompt - mongo                                          —   □   ×
> db.osmdata.aggregate( [ {$project: {'year': { $substr: [ '$created.timestamp', 0, 4 ] }}} ,
{$group: {_id:'$year', count:{$sum:1}}},{$sort:{count:-1}}, {$limit:1}] ).pretty()
{ "_id" : "2014", "count" : 171993 }
>
```

3. Additional Ideas

Improving postal code information

Postal codes always refer to one street and one (large) street can have multiple postal codes. Looking at 218 addresses with wrong postal codes, the majority of the wrong cases concerned postal codes which were used two times. Once for the correct street and once for the wrong street, which was often the adjacent street.

With the following aggregation it is easy to determine postal codes that are connected to two addresses.

```
> db.osmdata.aggregate( [ {$match: {'address.postcode': {$exists: true}}},
 {$group: {_id:'$address.postcode', street:{$addToSet:'$address.street'}}}
,{$unwind:'$street'},{$group: {_id:'$_id', count:{$sum:1}}},{$match: {'cou
nt': {$gt: 1}}}   ] )
{ "_id" : "4153AG", "count" : 2 }
{ "_id" : "4191KP", "count" : 2 }
{ "_id" : "4153AE", "count" : 2 }
{ "_id" : "4191KR", "count" : 2 }
{ "_id" : "4153AC", "count" : 2 }
{ "_id" : "4153AJ", "count" : 2 }
{ "_id" : "4153RX", "count" : 2 }
{ "_id" : "4158CN", "count" : 2 }
{ "_id" : "4153RS", "count" : 2 }
{ "_id" : "4158CH", "count" : 2 }
{ "_id" : "4191PN", "count" : 2 }
{ "_id" : "4191KH", "count" : 2 }
{ "_id" : "4116BB", "count" : 2 }
{ "_id" : "4191KJ", "count" : 2 }
{ "_id" : "4116EB", "count" : 2 }
{ "_id" : "4111RJ", "count" : 2 }
{ "_id" : "4191CP", "count" : 2 }
>
```

My advice is to look at these situations manually to determine the right postal code. Because I have seen that the postal code site uses a lot of abbreviations where the osm data does not (e.g. Dr. vs Doctor) a pair of human eyes can help making the right decision. Searching for a location using postal code and house number is very popular in the Netherlands, because by typing in 7 to 9 characters (postal code is 6 characters and then 1 to 3 characters for house number) you usually have the correct location. Also typing in a postal code is less error prone than typing in a street address. Solving the situation where a postal code-house number-combination results in an ambiguous result is therefore very beneficial.