Final Project : Data wrangling with MongoDB

OpenStreetMap Sample Project
Data Wrangling with MongoDB.

17 May 2015
Claus H. Rasmussen

chrasmussen@me.com

- - - - - - - - - - - - - - - - - -

Statement:
I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc. By including this in my email, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate.

- - - - - - - - - - - - - - - - - -

Problems encountered in the map

Postcodes and street names was the first data, I looked at in the data set, using python scripts, e.g. using the same approach as in “Lesson 6 - Case Study : OpenStreetMap Data. Iterating through Ways Tags (#5)”.

Postcodes.
When looking the postcodes, there are very few errors, only 6 documents that should be corrected.

Sort existing postcodes by count, descending

```
> db.testdata.aggregate( {"$match" : { 'address.postcode' :
{ "$exists" : 1 } } }, { "$group" : { "_id" : '$address.postcode',
"count" : { "$sum" : 1 } } }, { "$sort" : { '_id' : 1 } } )
```

Street names.

Street names in Denmark are not often abbreviated as they are in the US, so this was not an issue with these data. Another problem was obvious, though, as most street namesin the data set are not mapped to the ‘correct’ key, “street.address”, but have their name values in the “name” key. This misunderstanding is even more complicated by the fact, that when analysing for “type” = “way”, a lot of point features also shows up for all types of shops and public institutions, e.g. “IKEA” and “University”.

Query for erroneous “name” and “type” pair (example).

```
> db.testdata.find({"name" : "IKEA", "type" : "way”}).count()
2
```

So here there is some work to be done to get the real street names into the “street.names” key and also removing the “way” type from features, that are not streets/roads. If I had to do it, I would start out by getting a list of all the real street names from the municipality of Aarhus, and by means of a python script, then wrangle the data into place.

Data Overview

The OSM used here is covering the City of Aarhus, Jutland, Denmark, Europe (6.8 MB zipped, 125.9 MB as XML):
URL: https://s3.amazonaws.com/metro-extracts.mapzen.com/aarhus_denmark.osm.bz2

```
> db.testdata.dataSize()
181106896

> db.testdata.stats()
{
        "ns" : "osm.testdata",
        "count" : 439235,
        "size" : 181106896,
        "avgObjSize" : 412,
        "storageSize" : 243314688,
        "numExtents" : 13,
        "nindexes" : 1,
        "lastExtentSize" : 68579328,
        "paddingFactor" : 1,
        "systemFlags" : 1,
        "userFlags" : 1,
        "totalIndexSize" : 14258944,
        "indexSizes" : {
                "_id_" : 14258944
        },
        "ok" : 1
}

> db.testdata.find({"type":"node"}).count()
392218

> db.testdata.find({"type":"way"}).count()
47006

# Number of users who have contributed to the dataset.

> db.testdata.distinct( 'created.user' ).length
316
```

Top ten contributing users

```
> db.testdata.aggregate({ "$group" : { "_id" : '$created.user',
"count" : { "$sum" : 1 } } }, {"$sort" : { 'count' : -1 } },
{"$limit" : 10 } )
{ "_id" : "Flare", "count" : 88449 }
{ "_id" : "AWSbot", "count" : 76299 }
{ "_id" : "rasmusv", "count" : 66482 }
{ "_id" : "Hjart", "count" : 26256 }
{ "_id" : "MichaelVL", "count" : 20324 }
{ "_id" : "Freek", "count" : 18920 }
{ "_id" : "antonr", "count" : 17138 }
{ "_id" : "Bilbo-denmark", "count" : 15627 }
{ "_id" : "Jesper Mortensen", "count" : 13488 }
{ "_id" : "mantson", "count" : 11757 }
```

Additional exploration using MongoDB queries

Contributor statistics

Top user contribution percentage (‘Flare’) : 20.14%
Combined top five users contribution : 63.25%
Combined top 10 users contribution : 80.77%

Most user have one one or a few posts, so it seems as if most of the map has been created from existing sources, most likely by automated mapping routines.

Top ten appearing amenities.

```
> db.testdata.aggregate( {"$match" : { 'amenity' : { "$exists" :
1 } } }, { "$group" : { "_id" : '$amenity', "count" : { "$sum" :
1 } } }, { "$sort" : { 'count' : -1 } }, { "$limit" : 10 } )
{ "_id" : "parking", "count" : 1201 }
{ "_id" : "post_box", "count" : 118 }
{ "_id" : "bench", "count" : 117 }
{ "_id" : "school", "count" : 88 }
{ "_id" : "fast_food", "count" : 73 }
{ "_id" : "bicycle_parking", "count" : 70 }
{ "_id" : "fuel", "count" : 70 }
{ "_id" : "recycling", "count" : 64 }
{ "_id" : "restaurant", "count" : 60 }
{ "_id" : "place_of_worship", "count" : 54 }
```

Not surprisingly, the top five contributors are also among the users, who have created most features (apart from ‘jrjosephsen’):

Top five users, who have added ‘parking’ amenities.

```
> db.testdata.aggregate( {"$match" : { 'amenity' : { "$exists" :
```

```
1 }, 'amenity' : 'parking' } }, { "$group" : { "_id" :
'$created.user', "count" : { "$sum" : 1 } } }, { "$sort" :
{ 'count' : -1 } }, { "$limit" : 5 } )
{ "_id" : "Flare", "count" : 454 }
{ "_id" : "rasmusv", "count" : 236 }
{ "_id" : "Hjart", "count" : 79 }
{ "_id" : "jrjosephsen", "count" : 54 }
{ "_id" : "MichaelVL", "count" : 43 }

# The restaurants most popular cuisine is not even registered(?).
Here's something to work on for the local Tourist Agency!

> db.testdata.aggregate( {"$match" : { 'amenity' : { "$exists" :
1 }, 'amenity' : 'restaurant' } }, { "$group" : { "_id" :
'$cuisine', "count" : { "$sum" : 1 } } }, { "$sort" : { 'count' :
-1 } }, { "$limit" : 5 } )
{ "_id" : null, "count" : 32 }
{ "_id" : "chinese", "count" : 4 }
{ "_id" : "regional", "count" : 4 }
{ "_id" : "italian", "count" : 2 }
{ "_id" : "pizza", "count" : 2 }
```

Professionally, I work as Data Manager for the Danish National Road Agency (responsible for 'Areas and Equipment' on a nationwide level) and I have a specific interest in looking into the possibility of comparing the OpenStreetMap data with the official data set. This might help me find errors in my databases and hopefully I might also be able to harvest some of the data from OSM :-)
I'm sure that I haven't stopped exploring the possibilities that Python and MongoDB has shown me in this excellent course.