DATA WRANGLING WITH MONGO DB

Analyzing OpenStreetMaps data

By Nikita Barsukov:

nikita@barsukov.net

barsukov.n@gmail.com

Map Area: Kharkiv, Ukraine
https://www.openstreetmap.org/export#map=11/49.9784/36.3016

Table of Contents

Introduction

This text is a grading paper for Udacity's course "Data Wrangling with MongoDB". For this paper it was required to download an OpenStreetMaps dataset representing any part of the world, and perform a brief analysis of this dataset, according to given specifications. Minimum required size of an uncompressed dataset is 50 Mb.

I chose an area around Kharkiv, a large the city on the East of Ukraine, where I grew up. The dump is available on http://1drv.ms/1SnZXM3

I visualized some aspects of this dataset, made an overview of the dataset, tried to identify and, if possible, fix inconsistencies, and suggested ways to improve the quality of dataset. When I identified problems and directions for improvement, I also dataset for Kharkiv with two other data dumps for Stockholm and Copenhagen.

I crested quite a few amount of code to generate plots and figures in this paper. I used MongoDB for data wrangling, as required in this course. All of the queries in MongoDB were wrapped in Python scripts, using pymongo library. Plots were created using R programming language.

All the code is available on my GitHub repository, https://github.com/nikita-barsukov/data_wrangling.

1. Problems with the dataset

After downloading XML file from Open Street Maps and importing it to MongoDB, I noticed several problems, namely:

- Addresses need to be standardized (street names and postcodes)
- Addresses having not enough fields
- Problematic translations of names to English and Russian languages
- Broken URLs of websites

I'll address them further below.

Standardizing addresses

First problem with data from open street maps which is very common for all the geographical areas was address standardization. It is often a problem that street names are abbreviated in different way, post codes are not normalized etc.

Surprisingly such problems were rather rare in my dataset. For example, postcodes in Ukraine consist of five digits with no spaces. In addition to that all the postcodes in Kharkiv region start with digit 6, hence less chance of having 4-digit postcodes because of leading zeroes. Indeed, if we look at length of postcode field, we will see following counts:

Total distinct postcodes	**105**
Postcodes with 5 digits	102
Postcodes with 11 digits	3

When we look closer at postcodes with 11 digits, we will see that these are ranges of postcodes, like 61000-61499. They are given for Kharkiv itself, and for two other smaller towns near it. Thus there is no clear indication that we should further normalize postcodes in the dataset.

Street names are also rather well normalized. The canonical street names are given in Ukrainian, and indeed the last words in street names are mostly full non-abbreviated words in

Ukrainian like “street”, “road”, “way” etc. There are 209 distinct street names, and 188 of them have valid endings. However, some of them have this word at the beginning of the name, which doesn’t follow convention. Also in some cases full address with house number is given in this field.

Finally, there is additional problem with translating street names to Ukrainian. There is a bunch of street names given in Russian transliteration. Easiest way to fix some of these inconsistencies was to find values that contain correct word for street at the beginning of the string, and put them to the end of the string, like that:

```
from pymongo import MongoClient
import re
client = MongoClient("mongodb://localhost:27017")
coll_kh = client['osm']['kharkiv']

regx = re.compile("^вулиця", re.IGNORECASE)
bad_steet_names = coll_kh.find({'address.street': regx})
for str_name in bad_steet_names:
    record = coll_kh.find_one({'_id': str_name['_id']})
    s = record['address']['street']
    s = s.replace('вулиця ', '')
    s = s + ' вулиця'
    record['address']['street'] = s
    coll_kh.save(record)
```

The problem of incorrect or inconsistent translations is described in more detail in section below.

Too few address fields

Another problem that I saw with the dataset also deals with addresses. A lot of given addresses contained just street name or a postal code. Some of it was valid, for example small towns around Kharkiv have just a postcode. However, many nodes representing houses, or amenities often have just a house number without street name, postal code or other address data. Counts of nodes with address by number of address fields looks like this:

Table 1 – Address fields in OSM dataset for Kharkiv area

Number of address fields	Number of documents
1	663
2	145
3	95
4	36
5	2

As we can see most of the nodes with given address do not have a full address. Compare this with open street maps data for Stockholm:

Table 2- Address fields in OSM dataset for Stockholm area

Number of address fields	Number of documents
1	48
2	32
3	478
4	290
5	415
6	19

There is no straightforward way to address this particular issue since it has to do with incomplete data entry at first place.

Inconsistent translations

Another problematic area of dataset of Kharkiv area is the fact that many proper names are given in several languages, typically English, Ukrainian and Russian. It creates some inconsistency, with plenty of named elements having only name in Ukrainian or Russian without English translation in it.

Table 3 - Number of translated names in OSM dataset of Kharkiv area

Total elements with name	**10.127**
Elements without English name	5.310
Elements without Russian name	3.806

Convention in the data set for Kharkiv is that by default all the geographical names are given in Ukrainian. Names in all the other languages have postfix, ex. field with key `name` should contain a name in Ukrainian, while `name:ru` contains its translation to Russian.

It is rather common that the default names are given in Russian. For example, if we count occurrences of Russian letter ы in various name fields, which does not exist in Ukrainian, we'll get numbers like this:

```
> db.kharkiv.find({'name': {$regex: /ы/}}).count()
238
> db.kharkiv.find({'address.street': {$regex: /ы/}}).count()
689
```

Russian letters in field 'name' often refer to names of various cafes or shops in Russian. However, we can easily fix street names. All of the problematic street names refer to a large

market area, with street name being the name of that area in Russian language. Fixing it in Python is straightforward:

```
from pymongo import MongoClient
client = MongoClient("mongodb://localhost:27017")
coll_kh = client['osm']['kharkiv']

bad_translations = coll_kh.find({'address.street': 'Рынок Барабашово'})
for str_name in bad_translations:
    record = coll_kh.find_one({'_id': str_name['_id']})
    record['address']['street'] = 'Ринок Барабашова'
    coll_kh.save(record)
```

Broken URLS

Finally, there is another problem area. Some data points have website field, and as it happens so often, some of them are no longer active. Moreover, the site URLS are not normalized, many of them do not have http:// prefix, trailing slash is present in some cases, and absent in another.

Out of 221 documents with website only eight had broken links. This problem is in fact easy to fix simply by removing broken websites from document, like this:

```
from pymongo import MongoClient
import requests

client = MongoClient("mongodb://localhost:27017")
coll_kh = client['osm']['kharkiv']

websites = list(coll_kh.find({'website': {'$exists': True}},
{'website': 1, 'name': 1}))

for site in websites:
    url = site['website']
    if 'http://' not in url:
        if 'https://' in url:
            url = site['website']
        else:
            url = 'http://' + url
    try:
        r = requests.get(url)
    except:
        print(url, ' connection error')
        rec = coll_kh.find_one({'_id': site['_id']})
        del(rec['website'])
        coll_kh.save(rec)
        print('Website deleted')
        continue
```

2. Data Overview

After identifying problems with dataset and trying to fix some of them, let's calculate basic statistics for the data after cleaning.

File sizes:

map_kharkiv	179,5 Mb
map_kharkiv.json	260 Mb

Number of documents:

```
> db.kharkiv.count()
901917
```

Number of nodes:

```
> db.kharkiv.count({'type': 'node'})
> 766864
```

Number of ways:

```
> db.kharkiv.count({'type': 'way'})
> 134983
```

Number of unique users:

```
> db.kharkiv.distinct('created.user').length
> 550
```

Top 5 contributors:

```
> db.kharkiv.aggregate([{"$group":{"_id":"$created.user",
"count":{"$sum":1}}}, {'$sort': {'count':-1}}, {'$limit': 5}])
> { "_id" : "dima_ua", "count" : 235303 }
{ "_id" : "_sev", "count" : 123736 }
{ "_id" : "Rereader", "count" : 77624 }
{ "_id" : "Vort", "count" : 59236 }
{ "_id" : "Svargref", "count" : 36323 }
```

Number of users with only one contribution:

```
 > db.kharkiv.aggregate([{"$group":{"_id":"$created.user",
"count":{"$sum":1}}}, {"$group":{"_id":"$count",
"num_users":{"$sum":1}}}, {"$sort":{"_id":1}}, {"$limit":1}])
> { "_id" : 1, "num_users" : 97 }
```

3. Additional ideas.

Number of amenities:

```
> db.kharkiv.count({'amenity': {'$exists': 'true'}})
3432
```

Number of distinct amenities:

```
> db.kharkiv.distinct('amenity').length
75
```

Top 5 amenities:

```
> pipeline = [{'$match': {'amenity': {'$exists': true}}},
{"$group":{"_id":"$amenity", "count":{"$sum":1}}}, {'$sort': {'count':-
1}}, {'$limit': 5}]
[
	{
		"$match" : {
			"amenity" : {
				"$exists" : true
			}
		}
	},
	{
		"$group" : {
			"_id" : "$amenity",
			"count" : {
				"$sum" : 1
			}
		}
	},
	{
		"$sort" : {
			"count" : -1
		}
	},
	{
		"$limit" : 5
	}
]
> db.kharkiv.aggregate(pipeline)
{ "_id" : "parking", "count" : 563 }
{ "_id" : "fuel", "count" : 247 }
{ "_id" : "school", "count" : 237 }
{ "_id" : "pharmacy", "count" : 196 }
{ "_id" : "bank", "count" : 189 }
```

Let's see how the dataset was filled over the time.

Plot on Figure 1 below shows number of created elements in this dataset by month.

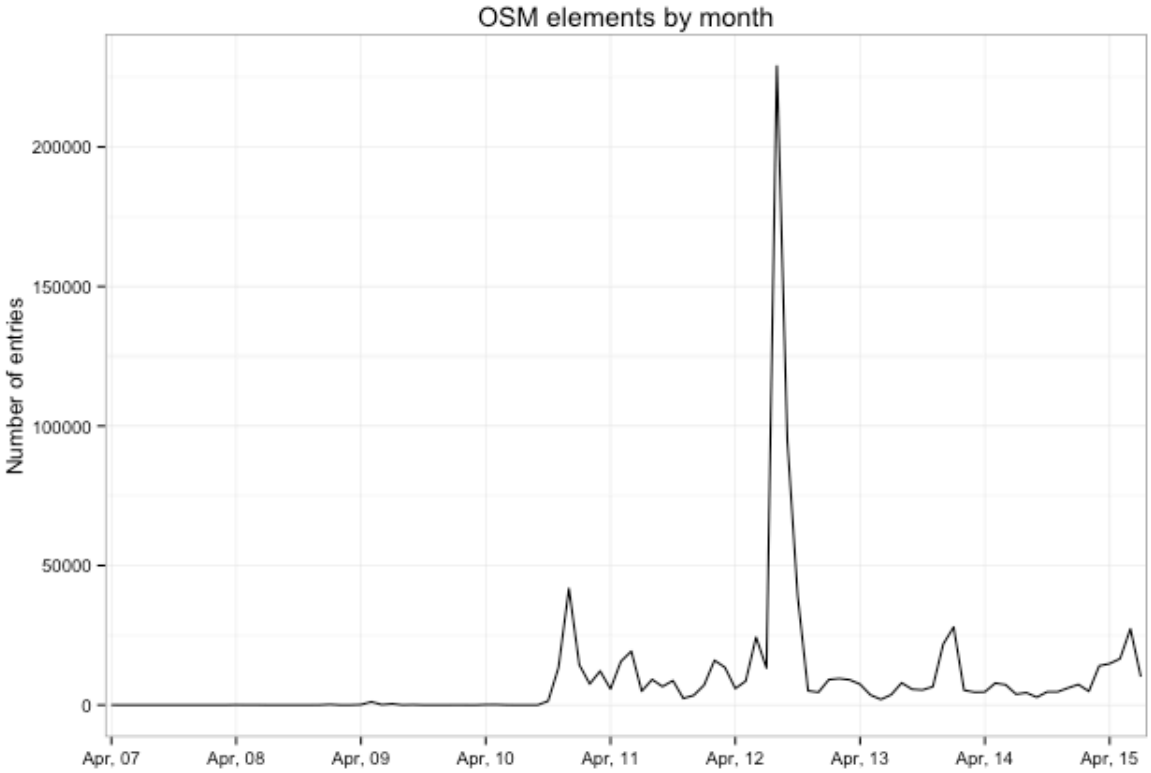

Figure 1

We can see that rate of adding elements to map of Kharkiv are is uneven. It was almost negligible during first three years, but it quickly started expanding after autumn 2010. We can also see a sharp spike in middle of 2012 (August 2012 to be precise). A bit less than 230.000 elements were created during this month, more than a quarter of all the elements in the dataset.

When we look at creation times at a different angle and break it down just by month and day of week, we will also see an interesting picture (see figure 2 below).

The breakdown by month shows us that most of entries were made in August, which corroborates with the previous line chart. However, breakdown by weekday is more or less even, there is no spike on that chart.

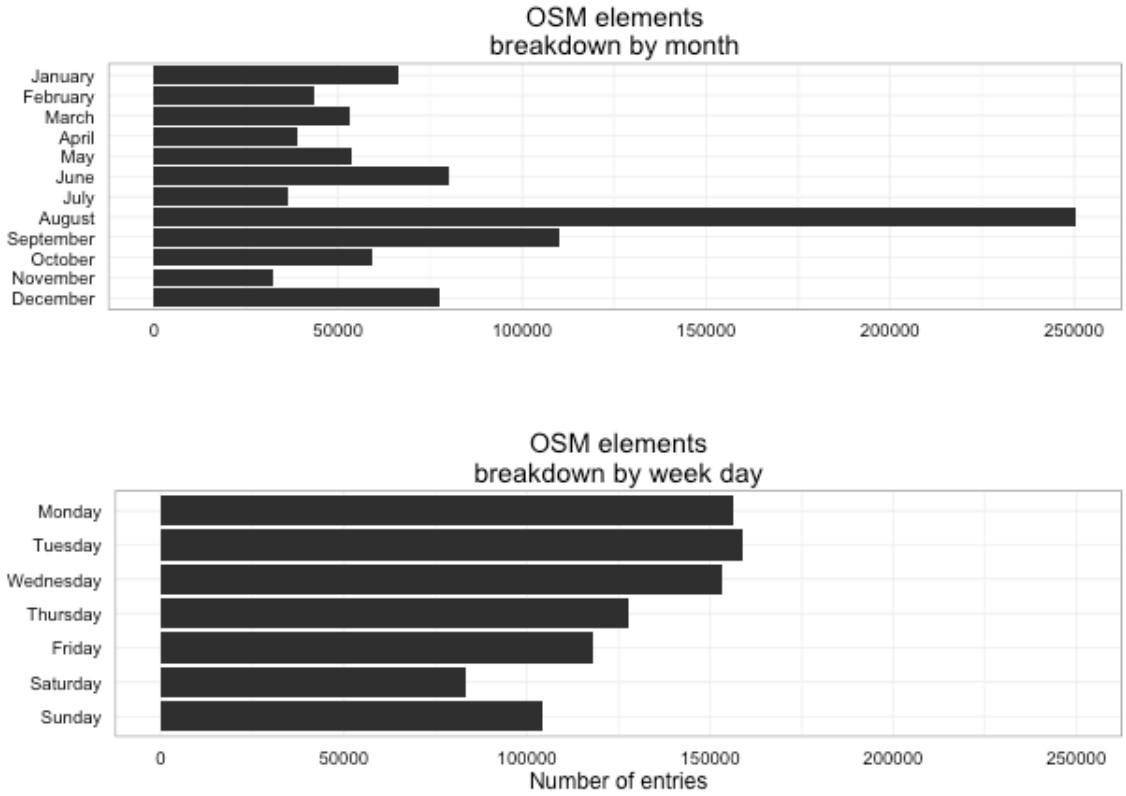

Figure 2

Improving dataset

The dataset can be improved in several different ways. An obvious thing would be to add more English variants of street names, amenities and other objects in the OSM database of the area. Also we could use some effort to fix inconsistent translations of names in Ukrainian.

Another area of improvement is to add better addresses, and I pointed out in previous section. There are too few amenities with web sites, it looks like more websites of banks, restaurants, companies etc could be added to the dataset.

Also it looks like there are too few cafes and restaurants in the OSM dataset for Kharkiv. In the dataset there are only 182 cafes and 98 restaurants, and note that Kharkiv's population is around 1,5 million. Compare this to Stockholm: 1371 restaurants and 747 cafes, or Copenhagen: 626 restaurants and 358 cafes. This can indicate that plenty of objects within Kharkiv are still not added to OSM.

We could add a gamification component as well. While it looks like the user contributions are not extremely skewed, what we could add is some incentive to add more and better translations and addresses.

Indeed, top contributor in my dataset is responsible only for about 26% of all the contributions, top 10 users created less than 75% of all the documents in the dataset. Thus what we could encourage is to add better and more complete translations and addresses, and more amenities in general. We could add leaderboards by number of cafes, offices, shops, and other similar object that seem to be under-represented in the dataset.

Also we could add leaderboards of those who added most translations to the names of various objects.

We can also add additional verification of coordinates of large geographical objects, like cities, towns or touristic attractions. We can use one of geo referencing APIs, like Google Maps Geolocation API, Bing Maps REST Services, MapBox Geocoding API or similar. We can query the names of such places from OSM database, get geographical coordinates and compare them with the ones present in OSM database.

Sources

Since both Python and MongoDB are new technologies for me, I used on-line documentation and code snippets extensively.

1. Python Software Foundation. Python Language Reference, version 3.4.2. Available at http://www.python.org
2. R Core Team (2015). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.
3. H. Wickham. ggplot2: elegant graphics for data analysis. Springer New York, 2009.
4. http://docs.mongodb.org/
5. http://www.cookbook-r.com/Graphs/
6. https://docs.python.org/
7. http://stackoverflow.com/questions/18337407/saving-utf-8-texts-in-json-dumps-as-utf8-not-as-u-escape-sequence
8. http://stackoverflow.com/questions/3996904/generate-random-integers-between-0-and-9
9. http://stackoverflow.com/questions/2824157/random-record-from-mongodb
10. http://stackoverflow.com/questions/15092884/how-can-i-return-an-array-of-mongodb-objects-in-pymongo-without-a-cursor-can
11. http://stackoverflow.com/questions/3086973/how-do-i-convert-this-list-of-dictionaries-to-a-csv-file-python
12. http://stackoverflow.com/questions/13281733/is-it-possible-to-flatten-mongodb-result-query
13. http://stackoverflow.com/questions/16406329/python-dictionary-count-of-unique-values
14. http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_%28ggplot2%29/
15. https://stat.ethz.ch/pipermail/r-help/2007-September/140509.html
16. https://jira.mongodb.org/browse/SERVER-6074