

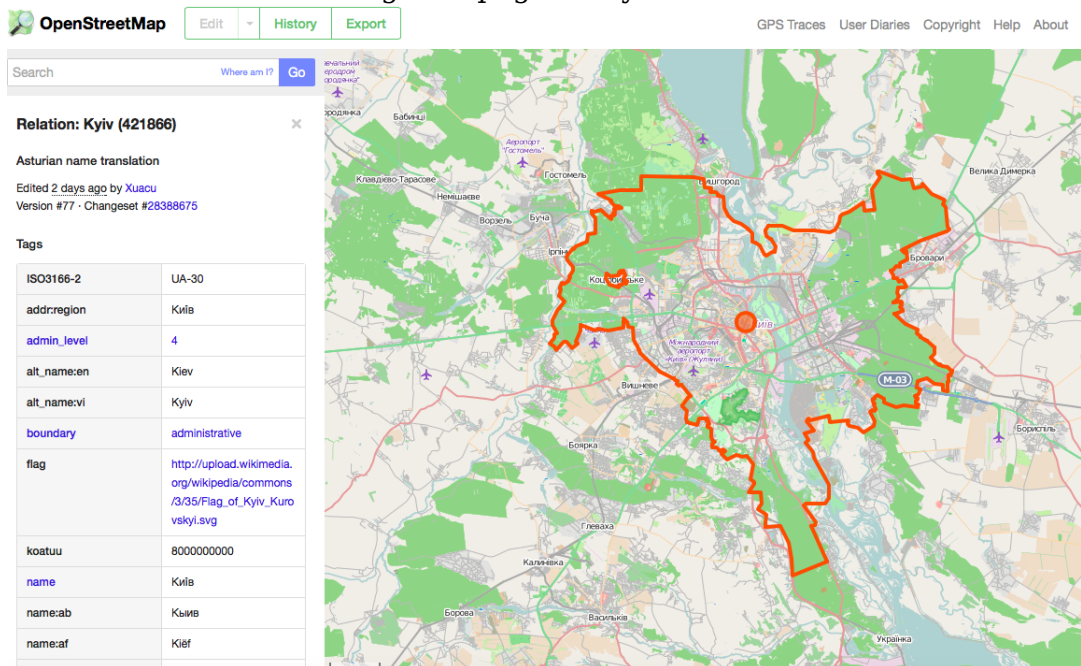
# Data Wrangle OpenStreetMaps Data

Oleksii Renov

February 3, 2015

## Choosing map area

Choosing map area is very simple for me. I live in Kiev, Ukraine. So i hadn't got another choice. As the very first step I need minimum and maximum longitude and latitude. For this task I'd gone to <http://openstreetmap.org> in the search field I'd typed Kiev and choose the second link. I've the following web page in my browser.



The screenshot shows the OpenStreetMap interface. On the left, a sidebar displays metadata for the relation 'Kyiv (421866)'. The sidebar includes a search bar, a 'Go' button, and a 'Where am I?' button. Below the search bar, the relation name is shown, followed by 'Asturian name translation', 'Edited 2 days ago by Xuacu', and 'Version #77 · Changeset #28388675'. A 'Tags' section contains a table with various tags and their values. The main map area shows a satellite view of Kyiv, Ukraine, with a red boundary outlining the city's administrative area. The map includes labels for various locations and features, such as 'Киев', 'М-03', and 'Україна'.

Tag	Value
ISO3166-2	UA-30
addr:region	Київ
admin_level	4
alt_name:en	Kiev
alt_name:vi	Kyiv
boundary	administrative
flag	<a href="http://upload.wikimedia.org/wikipedia/commons/3/35/Flag_of_Kyiv_Kurovskiy.svg">http://upload.wikimedia.org/wikipedia/commons/3/35/Flag_of_Kyiv_Kurovskiy.svg</a>
koatuu	800000000
name	Київ
name:ab	Кыив
name:af	Kiëf
...	...

I've done some zomming and choose next latitude and longitude parameters in my case:

- minimum latitude = 29.9048;
- maximum latitude = 50.0986;

- minimum longitude = 31.1600;
- maximum longitude = 50.7052.

Next step is to download my dataset. I've gone in [http://overpass-api.de/query\\_form.html](http://overpass-api.de/query_form.html), where i can past some queries to download data what i need.

My query is next:

```
(node(29.9048, 50.0986, 31.1600, 50.7052);<);out;
```

Anyway there is exists numbers of variants loading data. One of them is simply go to this url <http://overpass-api.de/api/map?bbox=29.9048,50.0986,31.1600,50.7052>. The xml file which I've already downloaded is very big, near 250mb. In my usual work i've chance working with xml, and files near 1mb is very hard to analyze. Let's move to processing this dataset.

## Process Dataset

As very first step, let's check what tags are in our dataset. Also would be helpful to check for attributes. Let's start from tags.

```
1 {'bounds': 1,
2  'member': 75124,
3  'meta': 1,
4  'nd': 1316743,
5  'node': 1079619,
6  'note': 1,
7  'osm': 1,
8  'relation': 5016,
9  'tag': 417122,
10 'way': 153102}
```

Next json is corresponded to tag attributes.

```
1 {'changeset': 1237737,
2  'generator': 1,
3  'id': 1237737,
4  'k': 417122,
5  'lat': 1079619,
6  'lon': 1079619,
```

```

7 'maxlat': 1,
8 'maxlon': 1,
9 'minlat': 1,
10 'minlon': 1,
11 'osm_base': 1,
12 'ref': 1391867,
13 'role': 75124,
14 'timestamp': 1237737,
15 'type': 75124,
16 'uid': 1237737,
17 'user': 1237737,
18 'v': 417122,
19 'version': 1237738}

```

First of all i would like to notice, that 'k' is key and 'v' is value. There are equal number of keys and values what is obvious. But it isn't clear what this keys are holding. I would like to explore more details and understand what keys and values have the most frequency in my dataset. In my opinion top 10 would be perfect for start.

```

1 {('surface', 9780),
2 ('amenity', 9919),
3 ('name:en', 10396),
4 ('name:uk', 10646),
5 ('name:ru', 12823),
6 ('addr:street', 14084),
7 ('name', 23862),
8 ('addr:housenumber', 30646),
9 ('highway', 67770),
10 ('building', 73142)}

```

There are few nice things here. Due to Ukraine history we have official street name in ukrainian, unofficial russian and english names. For convinience I'm going to analyze here only international names. Let's check some values for key 'amenity'. For me it's very interesting.

```

1 {('place_of_worship', 300),
2 ('bench', 331),
3 ('atm', 374),
4 ('kindergarten', 419),
5 ('fuel', 425),

```

```

6 ('restaurant', 514),
7 ('cafe', 520),
8 ('school', 543),
9 ('pharmacy', 566),
10 ('bank', 770),
11 ('parking', 1812)}

```

Let's try to analyze values, we've chosen three regular expressions to check text for lowercase, problem strings and others. Especially we would like to explore data with bad keys.

```

1 {'lower': 258569,
2 'lower_colon': 163,
3 'other': 157676,
4 'problemchars': 714}

```

Let's take a look at keys which values contain problem characters.

```

1 {'building:color': 2,
2 'contact:phone': 40,
3 'phone': 558,
4 'roof:colour': 19,
5 'wheelchair': 1}

```

There are exists more keys, but i want to point at not standartized phone numbers for this dataset. Let's look at them.

```

1 '+3-8-093-207-31-01, +3-8-067-838-47-64, +3-8-067-401-75-61'
2 '+380 44 251-34-34',
3 '+38(098)-109-5001',
4 '+380 44 490 20 30',
5 '+38 067 500 81 89',

```

Oh.. There are lots of phone numbers in different formats. I'm going to create one format for phone number for this dataset. It's can will be a list of phones even list can have only size equal to 1.

From wikipedia [http://en.wikipedia.org/wiki/Telephone\\_numbers\\_in\\_Ukraine](http://en.wikipedia.org/wiki/Telephone_numbers_in_Ukraine) i've read that typical Ukrainian phone numbers is:

- +380 xx xxx-xx-xx (general phone numbers);

I've written script for this preprocessing, before converting data in JSON and loading data to MongoDB. Next little python script handle different data formats and standartize them.

## Listing 1: Phone standartization

```

def convertPhone(phone):
    # First step i'm going to remove all non number characters
    phone = re.sub('\D', '', phone)
    if len(phone) == 10:
        phone = "+38" + phone
    elif len(phone) == 7:
        phone = "+38044" + phone
    elif len(phone) == 12:
        phone = "+" + phone
    elif len(phone) == 11:
        phone = "+3" + phone
    elif len(phone) == 9:
        phone = "+380" + phone

    if len(phone) == 13:
        phone = phone[0:4] + '_' + phone[4:6] + '_' + phone[6:9] +
            '_' + phone[9:11] + '-' + phone[11:13]
    return phone

```

Results of transformation:

- '+380(44)593-12-06' to '+380 44 593-12-06';
- '+380-44-599-60-82' to '+380 44 599-60-81';
- '3317730' to '+380 44 331-77-30'.

Nice results. Let's audit streets names. Ukraine has next common street types:

```

expected = ["вулиця", "бульвар", "проспект", "узвіз", "площа", "провулок", "шосе", "набережна", "тупик", "дорога", "проїзд", "шлях"]

```

And next mapping:

```

mapping = [ "вул.": "вулиця",
"пр.": "проспект",
"пл.": "площа",
'пров.': 'провулок'

```

] Next step is converting xml to json and loading data in mongodb. I've used data.py for creating json. Then I've started mongo shell navigate to folder which contain map.json. I've used mongoimport tool:

```

mongoimport -d test -c kiev map.json

```

## Data Overview

Let explore how number of documents in our db and it size.

Listing 2: Size and number of documents in db.

```
> show dbs
admin      (empty)
examples  0.078GB
local      0.078GB
osm        (empty)
test       0.953GB
> use test
switched to db test
> db.kiev.find().count()
1232721
```

Next step is to find number of 'ways', 'nodes' and uniques users.

Listing 3: Basic statistics

```
> db.kiev.find({"type":"node"}).count()
1079585
> db.kiev.find({"type":"way"}).count()
152949
> db.kiev.distinct('created.user').length
1303
```

Let find more interesting statistics: number of cafes, cinemas, banks and fuels.

```
> db.kiev.find({'amenity':'cafe'}).count()
520
> db.kiev.find({'amenity':'cinema'}).count()
47
> db.kiev.find({'amenity':'bank'}).count()
770
> db.kiev.find({'amenity':'fuel'}).count()
422
```

## Additional Data Overview

Let's find top 10 amenities.

```
> db.kiev.aggregate([{"$match":{"amenity":{"$exists":1}}}, {"$group":{"_id":"$amenity",
"count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":10}])
{ "_id" : "parking", "count" : 1784 }
{ "_id" : "bank", "count" : 770 }
{ "_id" : "pharmacy", "count" : 566 }
{ "_id" : "cafe", "count" : 520 }
{ "_id" : "school", "count" : 517 }
{ "_id" : "restaurant", "count" : 514 }
{ "_id" : "fuel", "count" : 422 }
{ "_id" : "kindergarten", "count" : 405 }
{ "_id" : "atm", "count" : 374 }
{ "_id" : "bench", "count" : 331 }
```

Let's explore to 10 banks in Kyiv.

```
> db.kiev.aggregate([{"$match":{"amenity":"bank"}},
{"$group":{"_id":"$name", "count":{"$sum":1}}},
{"$sort":{"count": -1}}, {"$limit": 10}])
{ "_id" : "OschadnyBank", "count" : 87 }
{ "_id" : "PrivatBank", "count" : 86 }
{ "_id" : "UkrSibBank", "count" : 51 }
{ "_id" : null, "count" : 50 }
{ "_id" : "Raifaisen Bank Aval", "count" : 21 }
{ "_id" : "UniCredit Bank", "count" : 16 }
{ "_id" : "PraveksBank", "count" : 16 }
{ "_id" : "CityCommerce Bank", "count" : 15 }
{ "_id" : "Sberbank Russia", "count" : 14 }
{ "_id" : "Delta Bank", "count" : 13 }
```

All religions in city based on openstreetmaps data.

```
> db.kiev.aggregate([{"$match":{"amenity":"place_of_worship"}},
{"$group":{"_id":"$religion", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":1}])
{ "_id" : "christian", "count" : 275 }
```

How many traffic signals in Kiev based on OSM data?

```
> db.kiev.find({"highway":"traffic_signals"}).count()
854
```

Let's find top 10 combinations of website and amenity.

```
> db.kiev.aggregate([{"$match":{"amenity":{"$exists":1}, "website":{"$exists":1}},
{"$group":{"_id":{"amenity": "$amenity", "website": "$website"}, "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":10}])
{ "_id" : { "amenity" : "bank", "website" : "www.oschadnybank.com" }, "count" : 64 }
{ "_id" : { "amenity" : "bank", "website" : "www.ukrsibbank.com" }, "count" : 38 }
{ "_id" : { "amenity" : "bank", "website" : "http://www.citycommercebank.com/" }, "count" : 16 }
{ "_id" : { "amenity" : "bank", "website" : "www.deltabank.com.ua" }, "count" : 10 }
{ "_id" : { "amenity" : "bank", "website" : "http://www.oschadnybank.com" }, "count" : 9 }
{ "_id" : { "amenity" : "bank", "website" : "www.piraeusbank.ua" }, "count" : 7 }
{ "_id" : { "amenity" : "bank", "website" : "http://www.ukrsibbank.com" }, "count" : 7 }
{ "_id" : { "amenity" : "fast_food", "website" : "www.pizza-celentano.com" }, "count" : 6 }
{ "_id" : { "amenity" : "bank", "website" : "http://www.procreditbank.com.ua/" }, "count" : 6 }
{ "_id" : { "amenity" : "fast_food", "website" : "http://www.mcdonalds.ua" }, "count" : 5 }
```

## Conclusion

After this review of the data it's obvious that the Kiev city area is incomplete, though I believe it has been well cleaned for the purposes of this exercise. It interests me to notice a fair amount of GPS data makes it into OpenStreetMap.org on account of users' efforts, whether by scripting a map editing bot or otherwise. I've cleaned and standardized phone numbers and street names, also i've found that also should be standartized websites info. For me is very surprising to see, that the

most frequent amenity is parking and bank (my hypothesis was shops or restaurant). I've found top ten banks in Kiev using OSM data. It's interesting to know that we 854 traffic signals in Kiev. This was great experience to work maps data.

### **List of Resources**

1. Openstreetmap. Url: <http://www.openstreetmap.org>
2. Overpass Api. Url: [http://wiki.openstreetmap.org/wiki/Overpass\\_API](http://wiki.openstreetmap.org/wiki/Overpass_API)
3. MongoDB Documentation. Url: <http://docs.mongodb.org/manual/core/document/>
4. Python Mongo Drivers. Url: <http://docs.mongodb.org/ecosystem/drivers/python/>
5. Unicode HowTo. Url: <https://docs.python.org/2/howto/unicode.html>