Project 3
OpenStreetMap Data Wrangling with MongoDB

Igor Buinyi, igor.buinyi@gmail.com

Map Area: Kyiv, Ukraine
Downloaded on August 19, 2015 from
https://s3.amazonaws.com/metro-extracts.mapzen.com/kyiv_ukraine.osm.bz2

This is an updated version of my report. Here I respond to some of the reviewer's comments. Please look at section 'Other Ideas About The Dataset'.

Initial Data Processing

1. I downloaded a preselected map in OSM XML format for Kyiv, Ukraine (Ukrainian name 'Київ'). I live in this city so I chose to process the OpenStreetMap data for it. The size of the uncompressed file is 244MB.
   It should be noted that the local language is Ukrainian. It uses the Cyrillic script which is very different from the Latin script of the English language. I was aware I had to deal with potential encoding issues.

2. Then I started to explore the data. I modified the iterative parsing script from Lesson 6 and calculated the number of different tags and their attributes.
   (please see the code in *P3_1_iterative_parsing.py*)

Table: the number of tags (shown in rows) with a particular attributes (shown in columns)

	_found	ref	role	type	changeset	id	lat	lon	timestamp	uid	user	version	k	v
bounds	1	-	-	-	-	-	-	-	-	-	-	-	-	-
member	66,634	66,634	66,634	66,634	-	-	-	-	-	-	-	-	-	-
nd	1,377,892	1,377,892	-	-	-	-	-	-	-	-	-	-	-	-
node	1,124,829	-	-	-	1,124,829	1,124,829	1,124,829	1,124,829	1,124,829	1,124,826	1,124,826	1,124,829	-	-
osm	1	-	-	-	-	-	-	-	1	-	-	1	-	-
relation	5,466	-	-	-	5,466	5,466	-	-	5,466	5,466	5,466	5,466	-	-
tag	461,318	-	-	-	-	-	-	-	-	-	-	-	461,318	461,318
way	165,060	-	-	-	165,060	165,060	-	-	165,060	165,060	165,060	165,060	-	-

The data is 100% valid in the sense that each tag has the full list of attributes. For example, I found 5,466 '*relation*' tags and each of them has the following list of attributes: *changeset, id, timestamp, uid, user, version.* So, data irregularities can be found in the tag values only, not in the tag structure.

3. One second level tag "k" values contains a special character.

```
<tag k="&#127;phone" v="+38 (044) 456-35-55;+38 (098) 757-85-85"/>
```

I will correct the key to "phone". There are no other special characters in potential keys for the data dictionary.

4. Then I looked at the data patterns for streets and other objects.

The street data is stored in *'way'* tags with the corresponding *'ref'* and *'tag'* subtags. Street name is stored in 'name' attribute of the corresponding 'tag' subtag:

```
<way   id="99025191"   version="6"   timestamp="2013-07-01T06:33:46Z"   changeset="16773448"
uid="721672" user="Movses">
       <nd ref="1145589112"/>
       ...
       <tag k="name" v="Деміївський провулок"/>
       <tag k="highway" v="residential"/>
       <tag k="name:en" v="Demiivskyi Lane"/>
       <tag k="name:ru" v="Демеевский переулок"/>
       <tag k="name:uk" v="Деміївський провулок"/>
```

The Ukrainian version of the street name is normally found in the 'name' key. Sometimes records for English and Russian versions are found in 'name:en' and 'name:ru'

However, 'way' tag can store information about other objects, for example, parks or even buildings. The corresponding entries have the following structure:

```
<way   id="99058342"   version="5"   timestamp="2012-11-14T15:40:06Z"   changeset="13872780"
uid="440812" user="dudka">
       <tag k="building" v="yes"/>
       <tag k="addr:street" v="Казимира Малевича вулиця"/>
       <tag k="building:levels" v="11"/>
       <tag k="addr:housenumber" v="37/41"/>


<way   id="99060143"   version="7"   timestamp="2014-11-14T12:35:17Z"   changeset="26775862"
uid="440812" user="dudka">
       <nd ref="1145953792"/>
       ...
       <tag k="name" v="Архітектурний ф-тет КНУБА"/>
       <tag k="amenity" v="university"/>
       <tag k="name:ru" v="Архитектурный ф-тет КНУСА"/>
       <tag k="building" v="yes"/>
       <tag k="addr:street" v="Преображенська вулиця"/>
       <tag k="addr:housenumber" v="2"/>
```

Here, the key for street name is 'addr:street'. Some entries have 'name' key, but it contains the object name, not the street address. So, the key 'name', depending on the context, can contain either street name or other information.

This fact is important to understand because of the following reasoning. In some street names the following spelling error occurs: instead of "Main street" the record contains "street Main" (for the sake of clarity, I use the corresponding English-language pattern). In other cases the ending word (e.g. "street") is in Russian, not Ukrainian language. We can easily see few such errors in 'addr:street' attribute. But finding all of them in 'name' attribute is much more difficult because the misspelled street names are hidden in the multitude of other irregular object names.

In contrast to the dataset for Chicago, explored in Lesson 6, there are no shortened versions of street types like "Rd." or "Blvrd." at the end of address string.

5. Before we transform the data into a dictionary, we need to ensure validity of street names in 'addr:street' attribute.
   (please see the code in *P3_2_checking street names.py*)

All endings are shown below. 50 out 14,769 records appear to be invalid. Thus, we need to replace them.

Ukrainian	English	Number of entries
вулиця	street	13,018
проспект	avenue	626
провулок	lane	517
бульвар	boulevard	348
тупик	deadlock	62
шосе	road	47
набережна	quay	37
площа	square	29
узвіз	ascent	20
дорога	road	9
проїзд	passage	6
OTHER		50
ALL		14769

Here is the mapping to be included in our data processing script:

```
mapping = { "Срибнокильская": "Срібнокільська вулиця",
"Sortuvalna str."   : "Сортувальна вулиця",
"вул. Антонова"     : "Антонова вулиця",
"ул. Бориспольская" : "Бориспільська вулиця",
"Ботанічна"  : "Ботанічна вулиця",
"Бучми"      : "Бучми вулиця",
"Большая Васильковская"    : "Велика Васильківська вулиця",
"вулиця Виставкова" : "Виставкова вулиця",
"пров. Вишневий"    : "Вишневий провулок",
"Декабристів": "Декабристів вулиця",
"Златоустовская"    : "Златоустівська вулиця",
"Котовского" : "Котовського вулиця",
"проспект Леся Курбаса"    : "Леся Курбаса вулиця",
"Леніна"     : "Леніна вулиця",
"Лучистая"   : "Лучиста вулиця",
"Лісківська" : "Лісківська",
"вул. Андрея Малышко"      : "Андрія Малишко вулиця",
"вул. Микільсько-Слобідська"      : "Микільсько-Слобідська назва",
"Набережная" : "Дніпровська набережна",
"Олійника"   : "Олійника вулиця",
"Червоного Орача"   : "Червоного Орача вулиця",
"Осенняя"    : "Осіння вулиця",
"проспект Перемоги" : "Перемоги проспект",
"С. Петлюри" : "Симона Петлюри вулиця",
"вул. Сагайдачного" : "Сагайдачного вулиця",
"вул. Григорія Сковороди"  : "Григорія Сковороди вулиця",
"Совхозная"  : "Радгоспна вулиця",
"проспект Героев Сталинграда"    : "Героїв Сталінграду проспект",
"Чернобыльская"     : "Чорнобильська вулиця",
"Чехова"     : "Чехова вулиця",
"Электротехническая"       : "Електротехнічна вулиця",
"жовтнева"   : "Жотнева вулиця",
"Днепровская набережная"   : "Дніпровська набережна",
"Площа Дружби народів"     : "Дружби Народів Площа",
"Московський п"     : "Московський проспект",
```

```
"Автодорожная улица"       : "Автодорожна вулиця",
"Погребський шлях"  : "Погребський Шлях вулиця"}
```

It turns out that the most of this error entries (with wrong schema) are in Russian. That is why I decided to look though all of the correct entries in order to check whether they are Ukrainian. There are total 1039 different street (including lanes, roads, avenues etc.) names and all of them are in Ukrainian language. So we do not need to correct them.

6. In a similar way we can ensure validity of street names in 'name' attribute. But, as described in Section 2, few records of interest are hidden in thousands of others correct records, all of them having nearly indistinguishable structure. That is why I decided not to search for the error records.
   At first, I wanted to create a new variable 'streetname' to store "correct" street names (i.e. for those values that end with a correct word for a street name like "street" or "avenue"). But then I looked at the data at found that not only streets have such names, but also other objects do. For example, we can find bus station "Main street" which should not be confused with the street name.
   That is why I will rely on other tags to check whether the entry contains information for a street. One of these tags is 'highway'.

7. The dataset describes a rectangular area near Kyiv, so it includes a few objects outside the city. Some objects have a special tag key 'addr:city'. If the city is not similar to Kyiv, I eliminate the corresponding data entry. I found total 7,924 such entries for elimination, please see the table below.

addr:city	count	What to do?
Київ	1254	Nothing
київ	10	Change to "Київ"
Киів	1	Change to "Київ"
Киев	40	Change to "Київ"
Kyiv	3	Change to "Київ"
Kiev	1	Change to "Київ"
OTHER	7924	Eliminate

8. Now I'm ready to form the dictionary in JSON format. Similarly to the exercise in Lesson 5, I am going to process the data into the list of dictionaries that look like this:

```
{
"id": "2406124091",
"type: "node",
"visible":"true",
"created": {
          "version":"2",
          "changeset":"17206049",
```

```
          "timestamp":"2013-08-03T16:43:42Z",
          "user":"linuxUser16",
          "uid":"1219059"
        },
"pos": [41.9757030, -87.6921867],
"address": {
          "housenumber": "5157",
          "postcode": "60625",
          "street": "North Lincoln Ave"
        },
"amenity": "restaurant",
"cuisine": "mexican",
"name": "La Cabana De Don Luis",
"phone": "1 (773)-271-5176"
}
```

The following are the rules for my data processing:

- ✓ include only 2 types of top level tags: "node" and "way"
- ✓ all attributes of "node" and "way" should be turned into regular key/value pairs, except:
  - attributes in the CREATED array should be added under a key "created"
  - attributes for latitude and longitude should be added to a "pos" array,
  - for use in geospatial indexing. Make sure the values inside "pos" array are floats and not strings.
- ✓ one second level tag "k" value with problematic characters is replaced with the correct value
- ✓ if second level tag "k" value starts with "addr:", it should be added to a dictionary "address"
- ✓ if second level tag "k" value does not start with "addr:", but contains ":", you can process it same as any other tag.
- ✓ if there is a second ":" that separates the type/direction of a street, the tag should be ignored
- ✓ 2 occurrences of 'addr:street_1' tag are ignored
- ✓ The data nodes with 'addr:city' attribute value outside Kyiv are ignored
- ✓ The attributes 'addr:city:en' are ignored
- ✓ If second level tag "k" value is "type", change it to "typeinner"

(Please see the code in *P3_3_data_wrangling.py*)

I wroted the data to JSON file with size of 279MB. Then I imported the date to MongoDB with the help of the following command:

```
mongoimport --db osmap --collection kyiv --drop --file D:\Udacity\IntroToMongoDB\kyiv_ukraine.json
```

Data Overview

(Please see the code in *P3_4_working with MongoDB.py*)

```
# Number of documents
    print(db.kyiv.find().count())
1281965

# Number of nodes
```

```
    print(db.kyiv.find({"type":"node"}).count())
1118200

# Number of ways
    print(db.kyiv.find({"type":"way"}).count())
163765
```

In the tag description we can find that *"The highway=* tag is used for roads, paths and cycletracks and other recognised routes on land"*. However, this is only half-truth as the tag can also store information about other very different objects like bus stops or street lamps. So, "way" here should not be interpreted as "road".

```
# Number of unique users
    print(len(db.kyiv.distinct("created.user")))
1442

# Top 10 contributing users
    pprint.pprint(list(db.kyiv.aggregate([#{"$match":{"created.user":{"$exists":True}}},
                        {"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                                        {"$sort":{"count":-1}},
                                        {"$limit":10}
                                        ])))
[{'_id': 'Freeways_me', 'count': 169699},
 {'_id': 'matvey_kiev_ua', 'count': 83428},
 {'_id': 'Kilkenni', 'count': 81275},
 {'_id': 'dudka', 'count': 59402},
 {'_id': 'kulyk', 'count': 47177},
 {'_id': 'Cabeleira', 'count': 44896},
 {'_id': 'Barbos', 'count': 38661},
 {'_id': '1gorok', 'count': 32774},
 {'_id': 'Roman_kuz', 'count': 28804},
 {'_id': 'Alex-7', 'count': 25678}]

# Total amenities
    print(db.kyiv.find({"amenity":{"$exists":True}}).count())
11460

# Unique amenity type
    print(len(db.kyiv.distinct("amenity",{"amenity":{"$exists":True}})))
125


# Top 10 amenity types
    pprint.pprint(list(db.kyiv.aggregate([{"$match":{"amenity":{"$exists":True}}},
                                        {"$group":{"_id":"$amenity", "count":{"$sum":1}}},
                                        {"$sort":{"count":-1}},
                                        {"$limit":10}
                                        ])))
[{'_id': 'parking', 'count': 1982},
 {'_id': 'bank', 'count': 859},
 {'_id': 'pharmacy', 'count': 672},
 {'_id': 'bench', 'count': 655},
 {'_id': 'cafe', 'count': 615},
 {'_id': 'restaurant', 'count': 580},
 {'_id': 'atm', 'count': 517},
 {'_id': 'school', 'count': 504},
 {'_id': 'kindergarten', 'count': 417},
 {'_id': 'fuel', 'count': 406}]
```

Assessing Data Quality

A goal of this project is to evaluate the data validity, accuracy, completeness, consistency, uniformity. Now I am going to reflect on those issues.

Validity

Validity can be measured as a degree to which entries in the dataset conform to a predefined schema.

As we found above, all tags in the downloaded file are valid since they contain all necessary attributes. Moreover, the information in the file seems to follow the specifications of OpenStreetMap. For example, such seemingly extravagant values of 'highway' tag as "street_lamp" or "speed_camera" do adhere to the specification. These arguments support the statement that the data is valid.

There is one tag key with a special character in the dataset:

```
<tag k="&#127;phone" v="+38 (044) 456-35-55;+38 (098) 757-85-85"/>
```

I corrected the problem, so it does not affect validity of keys.

So, I conclude that the data is valid [enough for us to work with it].

Accuracy

Accuracy measures how data conforms to the gold-standard data. In other words, it shows whether the information in the dataset is correct.

To measure accuracy, we need to compare the data with the gold standard. I do not have another source of map information that can be used programmatically.

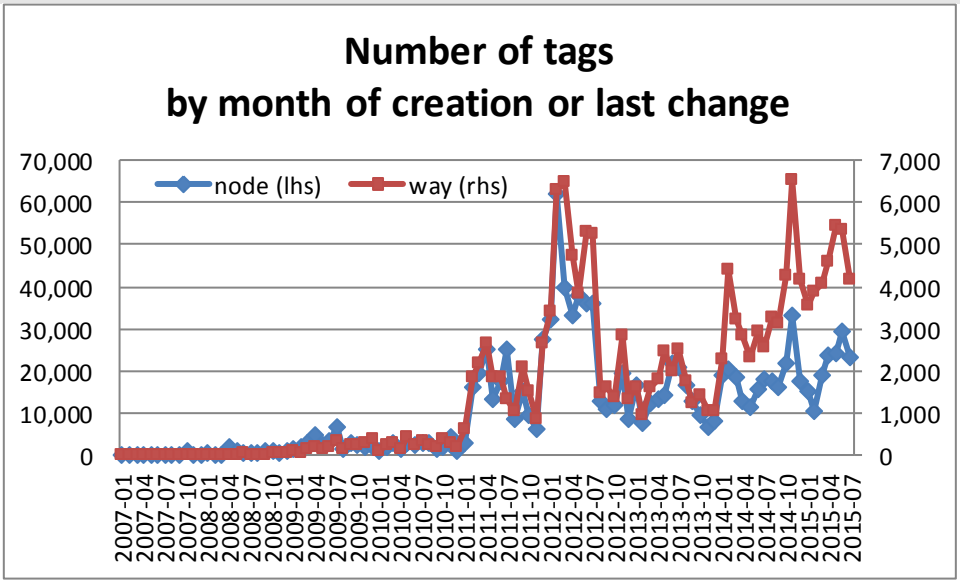
But I am going to use one indirect method to access accuracy. We know that the real data is not static, so we would expect many changes to occur in the recent months. If there were few little changes, we would suspect that the data is outdated and, therefore, inaccurate.

I executed the following query, transformed the data and then plotted it in Excel.

```
l=list(db.kyiv.aggregate([{"$match":{"created.timestamp":{"$exists":True}}},
                  {"$project": {'_id':0, "type":1, "timestamp":"$created.timestamp"}}
                          ]))
    res={}
    for item in l:
        month=item['timestamp'][0:7]
        if month not in res:
            res[month]={}
            res[month]['node']=0
            res[month]['way']=0

        res[month][item['type']]+=1
```

```
    pprint.pprint(res)
```



The number of recent changes is high, for both 'nodes' and 'ways'. So, in this test I did not found the evidence that the data is inaccurate.

Further checks would require comparing the downloaded data with the data from other sources, but it is going to be a too tough task, so I will not proceed with it.

Completeness

Completeness measures whether we have all information we should have.

Shannon Bradshow told us in his lecture that completeness is about how many **records** we have. If completeness is defined this way, it also requires an external source of data (or even gold-standard data) for precise estimate.

But completeness can be also understood as how much information within a record we actually have. In this case, our task becomes less complicated.

Let's start with street names.

As of January 2015, there were 2715 street names in Kyiv (source in Ukrainian http://kga.gov.ua/images/dovidnyk-vul/dovidnykvyl.pdf) including 1960 "streets" and 520 "lanes". In our dataset only 873 distinct street names in the 'addr:street' field.

```
# Number of unique street names
    print("Number of unique street names: ",len(db.kyiv.distinct("address.street")))
873
```

This is far below the real number. So, even if there are all records in the dataset, many of them lack the address information.

I also found how many streets in the dataset have 'name' tag. Remember that tag 'highway' is used to denote a street, but this tag is also used for other very different objects. So, I grouped the records by tag 'highway' and found how many of them have attribute 'name' .

```
pprint.pprint(list(db.kyiv.aggregate([{"$match":{"highway":{"$exists":True}}},
          {"$project": {'_id':0, "highway":1, "name":1,
                        "value":{"$ifNull": [ "$name", "_withoutname" ]}}},
          {"$project": {'_id':0, "highway":1, "name":1,
                        "value":{"$cond":[{"$eq":["$value","_withoutname"]},0,1]}}},
          {"$group":{"_id":"$highway", "found":{"$sum":1}, "with_name":{"$sum":"$value"}}},
          {"$sort":{"found":-1}},
          {"$limit":100}
           ])))
```

The results are listed in the following table. We can see that “residential”, “primary”, “secondary”, “tertiaty” and “trunk” roads have 46-79% of records with street names. They are likely to represent real streets.

'highway' tag	objects found	with name	with_name, %
service	19,139	160	1%
footway	14,422	27	0%
residential	10,260	4,764	46%
track	6,739	45	1%
crossing	3,440	0	0%
path	3,336	28	1%
steps	2,636	7	0%
unclassified	2,316	577	25%
bus_stop	1,974	1,362	69%
tertiary	1,449	994	69%
secondary	1,300	1,032	79%
traffic_signals	872	0	0%
primary	865	668	77%
trunk	624	385	62%
living_street	505	91	18%
trunk_link	368	20	5%
street_lamp	284	0	0%
primary_link	265	12	5%
pedestrian	200	37	19%
secondary_link	158	21	13%
OTHER	573	49	9%

Now I will find the number of unique street names in these 8 highlighted categories: residential, unclassified, primary, secondary, tertiary, living_street , trunk, pedestrian. If bus stops are excluded, those categories represent 96% of total named ‘highway’ objects in the dataset.

```
highway_list=["residential","unclassified","tertiary",
              "secondary","primary", "trunk","living_street", "pedestrian"]
# Unique street names:
print(len(db.kyiv.distinct("name",
                  {"name":{"$exists":True},"highway":{"$in":highway_list}})))
2692
```

Not bad! This is pretty close to the actual of 2715. The difference can be explained by (a) few errors in the street names (I checked, there are only few errors, indeed) and (b) the fact that I excluded non-highlighted categories from the search.

That is why we can make the following conclusion: although address fields are far from completed, there are really [almost] all streets in the dataset. It suggests that our dataset might have nearly all records, but those records lack much of the important information.

I believe the same consideration might apply to other objects. There are 672 pharmacies, 615 cafes, 580 restaurants and 517 ATMs are in the dataset. These are too small numbers for a 4-million city. So, only a small part of information about cafes, restaurants etc is in the dataset (low completeness), but it seems that nearly all building and streets are shown on the map (high completeness).

Consistency

Consistency indicates whether the fields that represent the same data in different records are the same.

I find it very cumbersome that the street names can be stored in 'addr:street' tag and 'name' tag. Moreover, in the 'name' tag street names can be confused with the names of other objects. I think, it is an example of inconsistent representation of street names in the dataset. Here I want to describe an example when a problem might occur. Assume the city government decided to change the name of a street. If all street names where stored in one field, we would simply replace the old name with the new name in that field. But in our case we should search across different fields and consider other information about whether the record really represents a street name instead of bus stop with the same name.

However, I could imagine particular reasons why the creators chose such a schema to represent the data. Once the schema is chosen, it should be followed for the validity to be ensured. I think that validity if far more important than consistency because we can potentially get a consistent dataset from a valid dataset, but not vice versa.

Uniformity

For the data to be uniform, all entries with a particular tag should have the same format and be represented in the same units. For example, the distance should be either in kilometers or miles in each entry in the uniform dataset. If some entries contain distance in kilometers and others in miles, the dataset is non-uniform.

I described how I corrected the street names in the 'addr:street' field. Now the 'address.street' values MongoDB are uniform.

In the section about completeness, I looked at the street names in the 'name' field. The street names are uniform except a few exceptions (I do not list the results here to make this report a bit shorter).

However, I found a few fields with non-uniform data.

i) Phone: a few examples listed below

```
<tag k="phone" v="+380 44 529 61 93"/>
<tag k="phone" v="0442855625,0442865403"/>
<tag k="phone" v="2536550"/>
```

This entries are not uniform and some of them do not have the necessary information (country code: second and third entry, city code: third entry). We can correct the value programmatically in order to get uniform and correct information.

ii) Opening_hours

```
<tag k="opening_hours" v="24/7"/>
<tag k="opening_hours" v="Mo-Fr 10:00-20:00; Sa-Su 10:00-18:00"/>
<tag k="opening_hours" v="mo-fr 09:00-19:00"/>
<tag k="opening_hours" v="10.00-22.00"/>
```

In the third entry the letters are lowercase, in the fourth entry the dot (.) is used instead of the colon (:) to split hours and minutes. In order to get uniform entries, we have to correct them.

Other Ideas About The Dataset

In this section I expand my reasoning in response to the reviewer's comments.

As can be seen from my analysis, it looks like the dataset have [almost] all the required entries but much of the important information about address or amenity type is missing. This might be a serious limitation for the potential uses of the dataset. For example, if someone is looking for a restaurant, she should be ready to find only a small part of them on the map.

To add the missing information, we might start a campaign targeted at cafes, restaurants and other small businesses to stimulate them to update their information on OpenStreetMap site. The details of the campaign might be the following: if a particular amenity updates their information, it is being promoted to a special category, let us name it "OpenStreetMap partners". OpenStreetMap will designate that the information about those organizations is verified. As a result, "partners" will get more traffic from OpenStreetMap, so they will have incentives to update their information. The "partners" will be advised to put at their premises certificates from OpenStreetMap indicating the completion of their profile on the site. It will improve the virality of the campaign. I believe that the cost of this campaign will be minimal since the response is expected to be viral, while the potential benefit will be large since all the major amenities will likely to update their information.

Another idea would be to stimulate more users to work on the updates. Particular gamification techniques can be applied. For instance, the standard gamification framework of points, badges and leaderboards should be introduced. The users will get point when they add useful information. If a user earns a particular amount of points, she gets promoted to another level. The best contributors can be seen in the leaderboard showing the best contributors.

The benefit of such arrangement is obvious: the users will have more incentives to work on the dataset. As a result, the quality of the dataset will improve and the dataset will become more useful for the customers. However, some of the contributors might see their goal as "to earn as many points as possible", not as "to improve the dataset in the best possible way". That is why the interests of the contributors and customers should be carefully aligned.

Another important idea is to use programmatic methods to ensure uniformity of the data. It will be relatively easy to implement a bot able to make phone numbers uniform and add country code '+38' and city code '044'

to them. In a similar way, the opening hours and other information including street addresses can be made uniform. With a limited investment, this will help to prevent errors for customers, thus improving the usability of the dataset.

Problems In The Dataset (Short Summary)

- Dataset contains information about objects outside Kyiv city – **fixed partly** (eliminated if the 'city' tag present)
- Special symbols in potential dictionary keys (one entry) – **fixed**
- Street names in 'addr:street' are not uniform – **fixed**
- Incomplete data on object addresses - many objects do not have address attribute – **not fixed**
- Phone numbers are non-uniform and incomplete (without country codes and city codes) – **not fixed,** can be fixed programmatically.
- Opening hours tags are not uniform – **not fixed,** can be fixed programmatically.

Conclusion

In this project I processed the downloaded data, cleaned it and imported to MongoDB. While working with the dataset, I found that the data is valid but incomplete and non-uniform. However, the dataset can be improved programmatically. For instance, phone numbers and opening hours can be transformed to a uniform schema with country codes and city codes. Our analysis also suggests that the dataset is likely to contain nearly all necessary records, but much of the important information within records is missing. For example, it seems few buildings have street address attribute. To improve the street address, we might combine the location data with the other source and load the address information to the dataset.