OpenStreetMap Project
Data Wrangling With MongoDB
Data Analyst NanoDegree

Mark Ayzenshtadt

Map Area: Moscow (South-Western part), Russia.
https://www.google.ru/maps/@55.6960753,37.545744,13z?hl=ru

1. Problems Encountered in the Map

The main problems encountered in the map are:

- Inconsistent house numbers
- Inconsistent phone numbers

Inconsistent house numbers.

There are several guidelines on house numbering:

- House numbers can sometimes have letters in it, e.g. 11a.
- If a house has several buildings, the building type and number or letter of the building is added.
- The building type can be either “корпус” (corpus) or “строение” (stroyenie), depending on whether or not it has a separate entry from the street, and it can also be a “владение” (vladeniye), if it is a custom-built structure or even not a building (e.g. construction site).
- In some cases, a house can have multiple building types and numbers: “14 к2с1”
- If a house stands on street intersection, it gets numbers from both streets, divided by a slash symbol, e.g. 5/16. The house can be referred by any of the streets (“Street_1 5/16” or “Street_2 16/5”).
- If a house was built in place of several demolished houses, its number is a range of the numbers of the demolished houses with a dash symbol: 7-11.

Everything above is only valid for Moscow, and in other cities houses are numbered differently.

To address the issue, and after no luck with using regular expression with Cyrillic characters, I’ve made a parser to deal with “housenumber” string character-by-character.

Inconsistent phone numbers.

In the OSM data, phone numbers can have various formats, for example: +7 495 1234567, +7-495-123-45-67, +7(495)123-45-67.

The external code of Russia (+7), can be substituted for 8, the internal code for calling outside your phone zone.

Objects can have several phone numbers, separated by either a dot or semicolon.

I chose to change numbers to the form of +7 495 12345678, separated by a comma if needed.

2. Data Overview.

The basic statistics of the dataset and the MongoDB Queries.

Size:

```
Moscow - SW.osm   ...   55.2 MB     #XML data
data.json         ...   60.3 MB     #non-cleaned JSON
data_cleaned.json ...   57.8 MB     #cleaned JSON
```

Number of documents: 261608

```
db.main.count()
```

Number of nodes: 217574

```
db.main.find({'type':'node'}).count()
```

Number of ways: 41391

```
db.main.find({'type':'way'}).count()
```

Number of unique users: 729
Top contributing user: Павел Гетманцев, 17928 entries.
Users, who made at least 1% of entries each: 22.
These users combined made 64% of the entries.
So, users, who made less than 1% of entries (less than 2616), together made 35% of the entries.
The 423 least contributing users combined made 1% of entries. Each of them made 25 entries or less.

```
user_data = db.main.aggregate([\
      {'$project': {'user':1, '_id':0}}, \
      {'$group' : {'_id' : '$user', 'count':{'$sum':1}}}, \
      {'$sort' : {'count': -1}}])
```

3. Additional MongoDB queries.

- Bus stops with/without shelter.
  This can be used to (very roughly) estimate weather conditions between regions.

```
db.main.aggregate([
            {'$match' : {'highway' : 'bus_stop', 'shelter' : {'$in' :
      ['yes','no']}}}, \
            {'$group' : {'_id' : '$shelter', 'count' : {'$sum' : 1}}}, \
            {'$sort' : {'count' : -1}} \
            ])
Bus stops with shelter:  206
Bus stops without shelter:  29
```

- Most popular leisure types

```
db.main.aggregate([ \
      {'$match' : {'leisure' : {'$exists' : 1}}}, \
      {'$group' : {'_id' : '$leisure', 'count': {'$sum': 1}}}, \
      {'$sort' : {'count' : -1}}, \
      {'$limit' : 5} \
      ])
{u'count': 519, u'_id': u'playground'}
{u'count': 447, u'_id': u'pitch'}
```

```
{u'count': 165, u'_id': u'park'}
{u'count': 49, u'_id': u'sports_centre'}
{u'count': 22, u'_id': u'track'}
```

- Most popular sports:

```
db.main.aggregate([ \
      {'$match' : {'sport' : {'$exists' : 1, '$ne' : 'multi'}}}, \
      {'$group' : {'_id' : '$sport', 'count': {'$sum': 1}}}, \
      {'$sort' : {'count' : -1}}, \
      {'$limit' : 5} \
      ])
{u'count': 50, u'_id': u'tennis'}
{u'count': 45, u'_id': u'soccer'}
{u'count': 25, u'_id': u'basketball'}
{u'count': 9, u'_id': u'volleyball'}
{u'count': 8, u'_id': u'football'}
```

- Most popular shop types:

```
db.main.aggregate([ \
      {'$match' : {'shop' : {'$exists' : 1}}}, \
      {'$group' : {'_id' : '$shop', 'count': {'$sum': 1}}}, \
      {'$sort' : {'count' : -1}}, \
      {'$limit' : 5} \
      ])
{u'count': 262, u'_id': u'convenience'}
{u'count': 148, u'_id': u'supermarket'}
{u'count': 85, u'_id': u'hairdresser'}
{u'count': 64, u'_id': u'car_repair'}
{u'count': 61, u'_id': u'florist'}
```

- Most popular cuisines:

```
db.main.aggregate([ \
      {'$match' : {'cuisine' : {'$exists' : 1}}}, \
      {'$group' : {'_id' : '$cuisine', 'count': {'$sum': 1}}}, \
      {'$sort' : {'count' : -1}}, \
      {'$limit' : 5} \
      ])
{u'count': 42, u'_id': u'coffee_shop'}
{u'count': 27, u'_id': u'italian'}
{u'count': 24, u'_id': u'burger'}
{u'count': 23, u'_id': u'japanese'}
{u'count': 18, u'_id': u'pizza'}
```

- Most common amenity types:

```
db.main.aggregate([ \
      {'$match' : {'amenity' : {'$exists' : 1}}}, \
      {'$group' : {'_id' : '$amenity', 'count': {'$sum': 1}}}, \
      {'$sort' : {'count' : -1}}, \
      {'$limit' : 5} \
      ])
{u'count': 658, u'_id': u'parking'}
{u'count': 281, u'_id': u'school'}
{u'count': 281, u'_id': u'kindergarten'}
{u'count': 264, u'_id': u'restaurant'}
{u'count': 255, u'_id': u'bench'}
```

4. Additional ideas on cleaning the dataset.

- **Generalizing street names**

This can be done by making a list of possible street types and splitting the street names into name and type.
In Russian, it is possible to change the order of the words in the street name (“Avenue of Lenin” and “Leninskiy Avenue” is the same street).

Thus, we must identify and correct objects that refer to the same street via different names.

Also, we need to assure (by nodes or coordinates) that “addr:street” tag of a building is actually the street it’s on.

- **Removing unconnected nodes**

This can be done by going through all the objects and finding nodes that do not belong to any.

- **Assuring that the data adheres to OSM guidelines**

This can be done by parsing the OSM wiki “Map Features” article.

5. Conclusion

User contribution:
While there are some users with high number of entries, likely through automated data processing, about 1/3 of the data is made by the users who made <1% entries each.
The 423 least contributing users (<25 entries each) together made 1% of entries (about 2600).
Thus, the distribution of user contributions has a rather long tail.

Data quality:
There are several flaws in this dataset that require some time to be cleaned, but the data on the area is reliable and very complete.
Depending on the project, the required level of cleanness of the data may vary, but even with a moderate amount of cleaning and processing, the data is extraordinary in terms of the possibilities it provides.