

Open Street Data Wrangling with MongoDB

Željko Veić

Map Area: Zagreb, Croatia

<https://www.openstreetmap.org/relation/3168167> (<https://www.openstreetmap.org/relation/3168167>)

<https://mapzen.com/data/metro-extracts> (<https://mapzen.com/data/metro-extracts>)

0. Choosing the Dataset

1. Problems Encountered in the Map

Abbreviated street names

No designation for a street type

Street type can be in various positions in the street name with variable capitalization

Numbers in place of street names

2. Data Overview

3. Additional Ideas

Contributor Statistics

Additional Data Exploration

4. Conclusion

5. References

0. Interest in the Dataset

I chose Zagreb dataset since it's the city I currently live in.

I was also interested to see how does parsing work when you have dataset in some language other than english.

Also, I wondered if there are some language-specific problems which might be encountered during the analysis.

1. Problems Encountered in the Map

In my data inspection I ran into several problems:

- Abbreviated street names
- No designation for a street type
- Street type can be in various positions in the street name with variable capitalization
- Numbers in place of street names

Abbreviated street names

There are street name entries which have street designation "Ul." (shortened from "Ulica" which means "Street" in translation).

Also, I found "Av." which is shortened from "Avenija" (translates into "Avenue")

No designation for a street type

Frequently you have street names without explicit designation of a street type ("Gajnice", "Jankomir", "Vijenac", ...).

In Croatia when you don't have street type specified the default street type is considered to be "Ulica" (i.e. "Street" in translation).

Street type can be in various positions in the street name with variable capitalization

Unlike the Chicago example from the course lessons where we could see that street type comes at the end of the street name, in Zagreb dataset we can see that is not the case here.

Street type "Ulica" can be at the beginning ("Ulica Ljudevita Gaja") as well as on the end of the street name ("Dalmatinska ulica").

Also note here that here street type "Ulica" is in one case capitalized and in other it's not.

In []:

```
{u'_id': u'Number of unique users', u'count': 569}
```

Top 1 contributing user

In []:

```
> db.locations.aggregate([{'$group': {'_id': '$created.user',  
                                     'count': {'$sum': 1}}},  
                          {'$sort': {'count': -1}},  
                          {'$limit': 1}])
```

In []:

```
{u'_id': u'Vedran V', u'count': 111202}
```

Top 10 users contribution sum

In []:

```
> db.locations.aggregate([{'$group': {'_id': '$created.user',  
                                     'count': {'$sum': 1}}},  
                          {'$sort': {'count': -1}},  
                          {'$limit': 10},  
                          {'$group': {'_id': 'Sum of top 10 users contributions',  
                                     'sum_all': {'$sum': '$count'}}}])
```

In []:

```
{u'_id': u'Sum of top 10 users contributions', u'sum_all': 355280}
```

Minimum number of contributions per user

In []:

```
> db.locations.aggregate([{'$group': {'_id': '$created.user',  
                                     'count': {'$sum': 1}}},  
                          {'$sort': {'count': 1}},  
                          {'$limit': 1}])
```

In []:

```
{u'_id': u'Aury88', u'count': 1}
```

Number of users having only three posts

In []:

```
> db.locations.aggregate([{'$group': {'_id': '$created.user',  
                                     'count': {'$sum': 1}}},  
                          {'$match': {'count': {'$eq': 3}}},  
                          {'$group': {'_id': 'Users having three posts',  
                                     'count': {'$sum': 1}}}])
```

In []:

```
{u'_id': u'Number of users having only one post', u'count': 155}
```

3. Additional Ideas

Contributor Statistics

Here are some user percentage statistics. We can see that very few users make up majority of all contributions:

Top user contribution percentage ("Vedran V") - **26,30%**

Combined top 3 users' contribution ("Vedran V", "Darko Boto" and "Janjko") - **60,43%**

Combined Top 10 users contribution - **84.03%**

Bottom 155 users which make up **27.24%** of all users contributed **0.0367%** of all entries.

Additional Data Exploration

Top 3 amenities

In []:

```
> db.locations.aggregate([{'$match':{'amenity':{'$exists': 1}},
                           {'$group':{'_id': '$amenity',
                                       'count': {'$sum': 1}}},
                           {'$sort':{'count': -1}},
                           {'$limit': 3}])
```

In []:

```
{u'_id': u'parking', u'count': 731}
{u'_id': u'cafe', u'count': 650}
{u'_id': u'restaurant', u'count': 336}
```

While I was looking at the type and number of amenities I thought it would be interesting to see this data used in some social or demographic studies.

It may answer some of the of the questions like:

- Do richer countries have more schools per citizen in their cities than poorer ones?
- Do cities with higher average population age have more restaurants than cafes?
- What can we say in general about the people living in a certain city just by analyzing it's infrastructure?
- etc.

Basically, how the town's infrastructure influences it's citizens and vice versa?

This knowledge would then in turn be useful to the policy makers when planning city development and investments.

It would be also useful to entrepreneurs who seek to expand their business to other cities. They could make more informed decision about which city or which neighbourhood within a city has the greatest potential to make their new branch successful.

The problems that might arise during this process might be unexpected answers to above questions. For example, data analysis might show that there is no significant correlation between the city infrastructure and people living there. Thus that differences between citizens of different cities arise from some other factors.

Most wide spread cafe chain

Since cafe is most frequent amenity type (behind the parking lot) let us see which chain is the most wide spread.

In []:

```
> db.locations.aggregate([{'$match':{'amenity':{'$exists': 1},
                           'amenity': 'cafe'}},
                           {'$group':{'_id': '$name',
                                       'count': {'$sum': 1}}},
                           {'$sort':{'count': -1}},
                           {'$limit': 3}])
```

In []:

```
{u'_id': None, u'count': 76}
{u'_id': u'Smart', u'count': 3}
{u'_id': u'Luna', u'count': 3}
```

It appears that "Smart" and "Luna" cafes are most wide spread.

However, there is a significant number of entries which do not contain the name of the object.

4. Conclusion

We can see that data for the city of Zagreb is actually very clean with very few errors and over abbreviations in the street name. Although there aren't any strict conventions on placing and capitalization of the street type name so replacing it properly does require some extra effort.

What's also visible is that very few users contributed to the great majority of the Zagreb open street map data.

Also, street type names in this case didn't contain any special characters but it would be interesting to see how the filtering works with some special chars in some other language perhaps.

It can also be seen on the cafe chain example that many of the names are missing and our dataset is incomplete.

5. Resources

1. Course Lessons
2. <http://stackoverflow.com/questions/6591931/getting-file-size-in-python> (<http://stackoverflow.com/questions/6591931/getting-file-size-in-python>)
3. <http://github.com/dunovank> (<http://github.com/dunovank>) - iPython notebook styling
4. <http://stackoverflow.com/questions/4022827/how-to-insert-some-string-in-the-given-string-at-given-index-in-python#4023434> (<http://stackoverflow.com/questions/4022827/how-to-insert-some-string-in-the-given-string-at-given-index-in-python#4023434>)
5. MongoDB documentation

Styling the iPython notebook:

In [22]:

```
from IPython.core.display import HTML
#HTML("<style>" + open("style.css", "r").read() + "</style>")
HTML("<style>{*background: #f8f8f8;}</style>")
```

Out[22]: