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. 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
- Numbers in place of street names
- Phone numbers come in multitude of formats

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).
The solution is to default **"type": "ulica"** to all instances which do not contain any of the official street types in their name.

Numbers in place of street names

There are entries in the dataset which have numbers as the street names ("3", "22", "49").
As the street namse In Croatia cannot be numbers this is obviously an error.
And since there are only three instances of this error in entire dataset we can remove them from the dataset completely.

Phone numbers come in multitude of formats

Phone numbers come in many diferent shapes and formats.
Preferred way of displaying numbers is "**+385**1234567" where the plus sign and first three digits are mandatory.
It should not contain any spaces, slashes or alphabetical characters (eg. "Tel: 00385 1 3324 506", "01/ 2990 072").
It should contain country and area codes and not be void of one or both of these (eg. "08007778")

2. Data Overview

Here are some basic statistics on the dataset and the MongoDB queries I used to attain them.

File size

In [26]:

```
import os

path_to_osm = '/home/veich/projects/nanodegree/003-open-street-map/dev/data/zagreb_croatia.osm'
path_to_json = '/home/veich/projects/nanodegree/003-open-street-map/dev/data/zagreb_croatia.osm.json'

print 'zagreb_croatia.osm ..... ' + str(os.path.getsize(path_to_osm) / (1000 * 1000)) + ' MB'
print 'zagreb_croatia.osm.json ..... ' + str(os.path.getsize(path_to_json) / (1000 * 1000)) + ' MB'
```

```
zagreb_croatia.osm ..... 81 MB
zagreb_croatia.osm.json ..... 89 MB
```

Number of documents

In [ ]:

```
> db.locations.find().count()
```

In [ ]:

```
422786
```

Number of nodes

In [ ]:

```
> db.char.find({"type": "node"}).count()
```

In [ ]:

```
357815
```

Number of ways

In [ ]:

```
> db.char.find({"type": "way"}).count()
```

In [ ]:

```
64971
```

Number of unique users

In [ ]:

```
> db.locations.aggregate([{'$match': {"created.user": {'$exists': 1}}},
                          {'$group': {'_id': '$created.user',
                                  'count': {'$sum': 1}}},
                          {'$group': {'_id': 'Number of unique users',
                                  'count': {'$sum': 1}}}])
```

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 betweeen 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 intetresting 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
6. http://stackoverflow.com/questions/1249388/how-do-we-remove-all-non-numeric-characters-from-a-string-in-python (http://stackoverflow.com/questions/1249388/how-do-we-remove-all-non-numeric-characters-from-a-string-in-python)

Styling the iPython notebook:

In [27]:

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

Out[27]:

In [ ]: