OpenStreetMap project

Data Wrangling with MongoDB

Łukasz Korona

Map area: a fragment of Kraków, Poland, including its historical districts
Min and max coordinates: [50.0359001, 19.8923] [50.088, 20.0051994]

1. Problems encountered in the map:

The sample that was downloaded by the author contained several issues, which needed to be dealt with during the cleaning phase. As the street names varied in the number of words and the frequency of appearing in the file, it was decided to process only street names with two, three or more words. Within each group, ten names with the highest frequency were corrected (in total thirty most frequent street names).

The first problem was connected with street names named after famous people. It is commonly accepted to use only last names while naming Polish streets. Using full names is inconvenient and sometimes impractical (e.g. when issuing official ID documents with limited space for address). In the .osm file, however, full names were used. To overcome this problem, parts of each audited street name that corresponded to the first name (checked against a regular expression pattern) were deleted. For example:

Józefa Dietla turns into *Dietla* as *Józefa* is a first name
Jana Zamoyskiego turns into *Zamoyskiego* as *Jana* is a first name

At the same time, certain street names (containing royal or the Pope's name) should not be changed, as they linguistically function as one element. Each street name was checked versus another regular expression containing historical names that should not be changed, even if they contain first names that are subject to processing. For example:

Aleja Jana Pawła II stays the same (the Pope's name)
Kazimierza Wielkiego stays the same; although *Kazimierza* is a first name, here we talk about a royal, historical figure

The second problem is similar to the first one – some street names contained military titles that had to be erased. Just like with the first problem, the street names were searched for patterns containing military titles (e.g. *marszałek, generał* etc.). If such a pattern was found, the fragment was erased. This was combined with the first problem, so that:

Aleja Marszałka Józefa Piłsudskiego turns into *Aleja Piłsudskiego* as *Józefa* is a first name, and *Marszałka* is a military title

The third problem was connected with Polish language special signs that are not covered by the popular ascii encoding. There are nine of them (ą, ę, ć, ś, ń, ł, ż, ź, ó) When printed with the print command, those signs appeared as they should have, e.g.:

```
print ‘Józefa’
```

gives:

```
Józefa
```

but in an output list patterns of strings replaced each problematic character, so that Józefa turned into [u'J\xf3zefa’]

This created discrepancies when trying to match street names against a predefined regular expression. In order to solve the problem, several steps were taken. First, the lists of street names to be processes were printed, for example, the two-part street names were printed as follows:

```
[(u'Kr\xf3lowej Jadwigi', 200), ('Juliusza Lea', 145), (u'Rynek G\u0142\xf3wny', 140), ('Waleregо Eljasza-Radzikowskiego', 113), ('Aleja Pokoju', 102), (u'J\xf3zefa Dietla', 102), ('Jana Zamoyskiego', 98), ('Kazimierza Wielkiego', 92), ('Henryka Wieniawskiego', 70), (u'Stanis\u0142awa Przybyszewskiego', 66)]
```

By matching the problematic characters with their substrings (for example ‘ó’ matches ‘\xf3), a regular expression could be built. In this expression, Polish characters were replaced with their corresponding substrings, for example (the letter *u* is for Unicode):

Józefa was used in a regular expression as *u’J\xf3zefa’*

By replacing problematic characters with their substrings, UnicodeWarning was ensured not to appear. As a result, all words from the file could be matched against the words and phrases from the predefined regular expressions.

2. Data overview

Below are summary statistics concerning the dataset that was prepared and uploaded into MongoDB in the collection “project”:

```
krakow2.osm……..95722 KB
krakow2.osm.json...102334 KB
```

The number of documents in the database:

```
>db.project.find().count()
457429
```

The number of nodes and ways:

```
>db.project.aggregate([{“$group”:{”_id”:”$type”, “count”:{“$sum”:1}}}])
{ “_id” : ”node”, “count” : 398247 }
{ “_id” : ”way”, “count” : 59182 }
```

The number of unique users:

```
>db.project.distinct(“created.user”).length
674
```

The top-contributing user:

```
>db.project.aggregate([{"$group":{"_id":"$created.user","count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":1}])
{ “_id” : “Władysław Komorek”, “count” : 235598 }
```

The number of shops:

```
>db.project.aggregate([{“$match”:{“shop”:{“$exists”:1}}}, {“$group:{“_id”:null, “count”:{“$sum”:1}}}])
{ “_id” : null, “count” : 1260 }
```

3. Additional ideas

3.1. Broadening the scope of the data cleaning – screen scraping of first names

One of the things that can be done in addition to what is already included in the project is the broadening of the regular expressions that are a referral list of what needs to be deleted, and what needs to stay within the street name. The manual processing of the street names can be long, for example, for the analyzed area:

```
>db.project.distinct(“address.street”).length
830
```

we find out that there are 830 of them, though we can assume many are one-worded. Among the two- or more worded street names, we will come across many first names that were not included in the regular expression in the project, because they appeared less frequently than the cleaned ones. Completing the expression for all street names, even within the analyzed area, might be a very mundane and time-consuming task.

The best way to extend the auditing process on all street names within the area (and also outside of it, on all of Kraków and even in other Polish cities) is to use screen scraping. The *Wiktionary* webpage provides the list of first names used in the Polish language:

https://pl.wiktionary.org/wiki/Indeks:Polski_-_Imiona

By entering sublinks in the link above, we can get the name in its correct form; for each name we would like to get its form in the "dopełniacz" grammatical case, one of seven cases in the Polish language. For the first name *"Agnieszka"*, we then get *"Agnieszki"*:

przypadek	liczba pojedyncza	liczba mnoga
mianownik	**Agnieszka**	**Agnieszki**
dopełniacz	Agnieszki	Agnieszek
celownik	Agnieszce	Agnieszkom

By going through all the first names in the *Wiktionary* list, we would be able to create a regular expression pattern for the first names almost automatically. There is one minor drawback to this idea, as not all first names have their own page, so they are listed in the *Wiktionary* link as red. Some of the first names, and thus street names, can be missed in the process of cleaning.

After screen scraping all the available first names, we would also have to check them for characters specific for the Polish language and process them so that they are in the correct form that does not bring up Unicode warnings.

Although the idea might not be perfect, it has got more pros than cons. Obviously, the biggest advantage of pulling the list of all first names automatically is that such a list (in the form a regular expression) can also be used to clean other Polish cities, as it is not uncommon to see streets named after famous people throughout Poland. As a result, what is built upon the analysis of one city can then be extrapolated with satisfying results on other Polish cities. Moreover, we reduce the problems connected with human errors, and we save time by not going through street names manually.

If this idea is combined with extending the second regular expression with royal or religious names (apart from the Pope, saints can also be included here), what we get is a fully-automated process of cleaning street names. Naturally, such a process would still need to see some revisions, to catch problems that were not thought of during the Python procedure building phase.

3.2. Additional data queried from MongoDB

The number of fuel stations (or other places that provide fuel):

```
>db.project.find({“amenity”:”fuel”}).count()
35
```

Which brand/company has got the biggest number of fuel stations:

```
>db.project.aggregate([{“$match”:{“amenity”:{“$exists”:1},
“amenity”:”fuel”}},{”$group”:{“_id”:”$brand”, “count”:{“$sum”:1}}},
{“$sort”:{“count”:-1}}])

{“_id” : null, “count” : 17 }
{“_id” : “Orlen”, “count” : 6 }
{“_id” : “BP”, “count” : 6 }
{“_id” : “Statoil”, “count” : 2 }
{“_id” : “Shell”, “count” : 2 }
{“_id” : “ARGE”, “count” : 1 }
{“_id” : “R8”, “count” : 1 }
```

(note: the most frequent fuel provider brand is labelled as null – filling in the missing amenity details might be a direction for future map updates)

How many places of worship are within the analyzed area:

```
>db.project.find({“amenity”:”place_of_worship”}).count
89
```

What types of places of worship are available within the analyzed area?

```
>db.project.aggregate([{“$match”:{“amenity”:{“$exists”:1},
“amenity”:”place_of_worship”}},{”$group”:{“_id”:”$building”,
“count”:{“$sum”:1}}}, {“$sort”:{“count”:-1}}])

{ “_id” : “church” , “count” : 65 }
{ “_id” : “synagogue” , “count” : 6 }
{ “_id” : “yes” , “count” : 4 }
{ “_id” : “chapel” , “count” : 4 }
{ “_id” : “null” , “count” : 3 }
{ “_id” : “basilica” , “count” : 3 }
{ “_id” : “cerkiew” , “count” : 2 }
{ “_id” : “cathedral” , “count” : 1 }
{ “_id” : “convent” , “count” : 1 }
```

We can see that the predominant place of worship within the analyzed area is a church, however, there is also half a dozen of synagogues, as well as a couple of Orthodox churches (cerkiew) and one Protestant convent. An additional idea is to clean the data with regards to “yes” and “null” records that can be seen in the output above. These are not valid terms for places of worship.

4. Conclusions

The Kraków area has been taken care of by OSM users with great care, however, it does not mean that no errors are produced. The main assumption about cleaning the data stated that full street names are inconvenient for bureaucratic purposes, so they need to be trimmed down, whenever possible (e.g. by stripping first names or military titles in case of famous people).

Additional aggregations have also been calculated, pointing to further possibilities of cleaning the data: some amenity information is missing or incorrect. Aside from that, we should also be cautious about characters that are specific for the Polish language.