Wrangling OpenStreetmapData with MongoDB

Selected map area: Area of Budapest, Hungary

https://www.openstreetmap.org/relation/22719
https://mapzen.com/data/metro-extracts/metro/budapest_hungary/

Table of contents

1. Problems Encountered in the Map

My first problem was that we can't find so many big cities in Hungary. :) My birth town (Gyula) and the place where I currently live (Szeged) was accessible in the database, but their size was well below the necessary 50MB.

My next try was the broader region where I live (Southern Great Plain in Hungary). It was large enough (~2GB uncompressed), but this area lays along the country border, so there were numerous street names from Serbia and Romania (what I don't understand). Finally, I chose the area of Budapest, the capital of Hungary.

To assess the clarity of the data I listed and collected user defined fields from a subsample of the whole dataset (~10% of the original).

First, I checked the country codes. I found where any country code was given it was correct and belongs to Hungary.
Second, I checked if is there any typo, abbreviation of the name of any city of the cities, but I found everything correct here.

After the mentioned audit I found the following potential errors:

- Inconsistent house numbers (17, 21B, 21/B, 21b ...).
- Inconsistent postal code (1023 vs. H-1012)
- Varying capitalization of street types (utca vs. Utca)
- Inconsistent phone number format

Inconsistent postal code

The postal codes in Hungary consists of 4 digits. Sometimes (especially in international mail delivery) they write as H-1213 (where H denotes the country). In all cases except one only the number were given, so I removed the leading "H-" from that single entry.

Varying capitalization of street types

The street types were surprisingly correct in my sample, only a couple of times was some capitalization issue. The issue what I faced with was the many different terms used for public places: utca (street), út (way), körút (boulevard), fasor (alley), tér (square) to name a few. I converted each name to lower case letter, removed the trailing dot and mapped abbreviation to the whole form.

Inconsistent house numbers

The basic house number is a single decimal number. If a land was divided, they put a / after the number and then a single letter like: 17/A, 17/B. Rarely, they denote divided plots not with letter, but with number, like 17/1-2. In my dataset I found various different formats: lower case letter and/or without slash, dashes etc. I transformed all of them to the standard format. In a small fraction of the house numbers I couldn't convert it because I don't even know what could be the real value. In this cases I removed the whole address tag.

Inconsistent phone number format

Phone numbers in Hungary has the following pattern:

- begins with country number (+36)
- followed by a 2 digit number (area code)
- then followed by 7 digits (for mobile phones) or by 6 digits (landlines), except in Budapest, because the area code is 1, followed by 7 digits.

Phone numbers can be written in various formats e.g.:

- +36201234567
- +36 20 123 4567
- +36 (20) 123-4567
- +36/20/1234567
- 0620123456 (06 is the internal country calling code)
- 003620123456 (international calling code with old format)

For simplicity, I transformed each of them to the standard format (without space, dashes, brackets...). However, there was a small portion of data which was erroneous: too many/few numbers, invalid area code, inconsistent area code and phone number. I dropped these items.

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

File sizes

```
budapest.osm ........499M
budapest.osm.json ...538M
```

Number of documents, nodes and ways

```
> db.budapest.find().count()
2502569
> db.budapest.find({"type":"node"}).count()
2119666
> db.budapest.find({"type":"way"}).count()
382754
```

Number of unique users

```
> db.budapest.distinct("created.user").length
2399
```

Statistics about places

```
Number of different cities
> db.budapest.distinct("address.city").length
69
```

3. Additional Ideas

3.1 Improvement ideas

I think that from the three issues what I mentioned in Chapter 1 standardizing phone numbers would have the best cost-effect ratio. Because, phone numbers can be formalized easily, maybe a real time checker could warn user to follow the convention.

Street names are very diverse by nature, so there we can't make such a big improvement unless we reform naming conventions.

I found one interesting thing. The most entry belong to Budapest, which is normal and expected (especially because Hungary is very centralized). But, the second most entry in this region came from Érd. This is a small town near to Budapest with 65000 citizen (compared to the population of Budapest which is around 2 million). I don't know the reason, why this happened.

I remember that 5 years ago, wikipedia announced a "Wikipedia loves monuments" photo contest, where people had to take photos of not so popular, but interesting sculptures, buildings. I think some contest could motivate users to contribute to the database.

Another idea for collecting routes and ways. I like to run and I usually use some runner app to track my activity. It would be great to extract these routes and submit to the database.

3.2 Possible benefits and problems of improvements

Standardizing phone numbers would result in the opportunity of faster searching. Sometimes, when I want to find a phone number whom it belongs to, I can't find any result in Google. I had to type in and try each possible combination to get result. Instead, if numbers would be in a unified format, searching, pattern matching, crawling would be easier and faster.

I can't think of any problem related to except one: the human factor. Many people are lazy and if the system would rejects his/her form because of the phone number checker, change their mind and don't submit any entry.

The benefits of my another suggestion are clear: if we organize a contest, hopefully much data will be submitted and the map will be larger and more accurate. But, I think the same people will be interested in this contest who edits the map now. So, the imbalance of the accuracy of the map can remain.

4. Additional data exploration using MongoDB queries

Cities with TOP10 entries

```
> db.budapest.aggregate([
  {
    "$match": {"address.city": {"$exists": true, "$ne" : null}}
  },
  {
    "$group": {_id:"$address.city", count:{$sum:1}}
  },
  {
    "$sort": {"count":-1}
  },
  {
    "$limit":10
  }])

{ "_id" : "Budapest", "count" : 48329 }
{ "_id" : "Érd", "count" : 25637 }
{ "_id" : "Törökbálint", "count" : 3263 }
{ "_id" : "Tárnok", "count" : 2868 }
{ "_id" : "Gödöllő", "count" : 1953 }
{ "_id" : "Pomáz", "count" : 1541 }
{ "_id" : "Sóskút", "count" : 1009 }
{ "_id" : "Halásztelek", "count" : 810 }
{ "_id" : "Nagykovácsi", "count" : 735 }
{ "_id" : "Budajenő", "count" : 554 }
```

Top 5 amenities

```
> db.budapest.aggregate([
  {
    "$match": {"amenity": {"$exists": true, "$ne": null}}
  },
  {
    "$group": {_id:"$amenity", count:{$sum:1}}
  },
  {
    "$sort": {"count":-1}
  },
  {
    "$limit":5}])
```

```
{ "_id" : "parking", "count" : 3360 }
{ "_id" : "bench", "count" : 2717 }
{ "_id" : "restaurant", "count" : 1547 }
{ "_id" : "waste_basket", "count" : 1338 }
{ "_id" : "bicycle_parking", "count" : 1243 }
```

Restaurant grouped by cuisine with more than 10 entries

```
> db.budapest.aggregate([
  {
    "$match": {
       "$and": [
        {"amenity":"restaurant"},
        {"cuisine":{"$exists":true,"$ne":null}}]
    }
  },
  {
    "$group": {_id:"$cuisine", count:{"$sum":1}}}, {"$sort":{"count":-1}},
    {"$match": {"count": {"$gte":10}}}
  ])

{ "_id" : "regional", "count" : 118 }
{ "_id" : "hungarian", "count" : 66 }
{ "_id" : "italian", "count" : 63 }
{ "_id" : "pizza", "count" : 61 }
{ "_id" : "chinese", "count" : 41 }
{ "_id" : "international", "count" : 32 }
{ "_id" : "indian", "count" : 23 }
{ "_id" : "burger", "count" : 19 }
{ "_id" : "asian", "count" : 10 }
{ "_id" : "turkish", "count" : 10 }
```

Top 10 entry with multiple phone numbers

```
> db.budapest.aggregate([
  {
    "$match": {
      "phone":{"$exists":true}
    }
  },
  {
    "$unwind":"$phone"
  },
  {
    "$group": {
      _id: "$name",
      count:{"$sum":1}
    }
  },
  {
    "$sort":{"count":-1}
  },
  {
    "$limit":10
  }])

{ "_id" : "SPAR", "count" : 16 }
{ "_id" : "Nyilvános telefon", "count" : 9 }
{ "_id" : "OTP Bank", "count" : 9 }
{ "_id" : "Shell", "count" : 7 }
{ "_id" : "Copyguru", "count" : 7 }
{ "_id" : "SPAR Szupermarket", "count" : 6 }
{ "_id" : "MOL", "count" : 6 }
{ "_id" : "Burger King", "count" : 6 }
{ "_id" : "Nyilvános távbeszélő állomás", "count" : 5 }
{ "_id" : "Friss Pékség", "count" : 5 }
```