Data Analysis Nanodegree – Project 3
Data Wrangling with MongoDB using OpenStreetMap dataset

Alexander Nenkov

Map area: Sofia, Bulgaria

http://www.openstreetmap.org/relation/1739543
https://s3.amazonaws.com/metro-extracts.mapzen.com/sofia_bulgaria.osm.bz2

1. Problems encountered in the map

After downloading the file, I started to research it programmatically with the mapparser.py file. The first thing I did was to check if the structure is ok according to OSM XML documentation. Fortunately, it was and I could continue to investigate the data without having to worry about the structure. There was several issues with the data which had to be corrected before inserting it as MongoDB documents:

- The postcodes were not matching the Sofia postcodes
- The street names were in Cyrillic
- The addr:street values were inconsistent
- There were incorrect streets

N.B. Throughout the document wherever is written that something is done "manually", what I mean is it was done programmatically but with direct matching for ex.: {"bul. Tzarigradsko Shose": "бул. Цариградско Шосе"}

Incorrect postcodes

The pattern for Sofia postcodes is quite easy to apply. They should be in the range [1000, 2000). There were 209 postcodes valued with "FIXME" or "fixme". The only solution here was to set them to None and skip them for the final documents.
There were also 57 postcodes not matching Sofia at all. After further investigating them it appeared that most of them were 2xxx which is a town near Sofia but there were also some from other countries in the world. The solution I took is to skip those documents. After standardizing them I could freely make aggregations on them:

```
> db.osm_data.aggregate([
   {"$match":{"address.postcode":{"$exists":1}}},
   {"$group":{"_id":"$address.postcode", "count":{"$sum":1}}},
   {"$sort": {"count": -1}}])
```

The top results for the query are:

[{"_id": 1000, "count": 498}, {"_id": 1113, "count": 384},
 {"_id": 1124, "count": 156}, ……]

No surprise here for the first one is as expected because this is the central city post office and people use it whenever they don't know the area post code.

The street names were in Bulgarian using Cyrillic

This was not an issue in understanding but rather in programming for it. At first I was not sure which way to go – English or Bulgarian names. But after investigation I found out that most of the names are in Bulgarian so I took that route. This enforced usage of Unicode functions throughout the code and was quite a challenge for the regular expressions. There were a total of 18 street streets with Latin characters which were matched to their Bulgarian translations manually with a matching dictionary.

Street values were inconsistent

There were many values with the same name but with or without street type street or different placement of the prefix "str.Name", "Str. Name, "str. Name" (бул. Име – in Bulgarian). I decided to make them all "str. Name". This was achieved with the following regex.

```
search=[
   unicode("бул", "utf-8"),
   unicode("ул", "utf-8"),
   unicode("пл.", "utf-8"),
]
re.compile('(' + "|".join(search) + ')(\s\.|\.\s|\.|\s)(?=\S)', re.UNICODE | re.IGNORECASE)
```

Additionally, there were lots of values with wrong capitalization. They were fixed by grouping them by lower() and deciding manually which is the correct version.

Incorrect street values

There were quite a few incorrect values. Some had quotes (5 different types) which was fixed with a regex, others were outright wrong – 2 web addresses, some "no", "c", an "apartments", etc. Most of these were dealt case by case but there is still a lot to do on this topic because individual values have to be fixed. For example: "Georgi S. Rakovski" vs "Georgi Sava Rakovski"

I was quite interested in the most represented street and after fixing the names, or at least most of them, here is the result:

```
> db.osm_data.aggregate([
   {"$match":{"address.street":{"$exists":1}}},
   {"$group":{"_id":"$address.street", "count":{"$sum":1}}},
   {"$sort": {"count": -1}}
])
```

```
[{"_id": "бул. Витоша", "count": 126}, {"_id": "бул. Васил Левски", "count": 80},
  {"_id": "бул. Патриарх Евтимий", "count": 62"},
 {"_id": "ул. Княз Борис I", "count": 61},
{ "_id": "ул. Каймакчалан",  "count": 55 }, …… ]
```

What struck me here was the 5th result which is a street I never heard of. It turned out that indeed it is a very small street but is located next to a shopping center and probably a lot of businesses used it as a marker.

2. Data Overview

sofia_bulgaria.osm ……………… 179 MB
sofia_bulgaria.osm.json ……… 207 MB

```
# Number of documents

> db.osm_data.find().count()
926436

# Number of nodes

> db. osm_data.find({"type":"node"}).count()
810749

# Number of ways

> db. osm_data.find({"type":"way"}).count()
115687
```

```
# Number of unique users

> db.osm_data.distinct("user.id").length
992

# Top 1 contributing user

> db.osm_data.aggregate([{"$group":{"_id":"$user.id",
"count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":1}])
[{"_id": "417398",  "count": 248647}]

# Top 3 amenities

[{"_id": "restaurant", "count": 457}, {"_id": "drinking_water", "count": 324},
   {"_id": "bank", "count": 251}]
```

3. Additional ideas

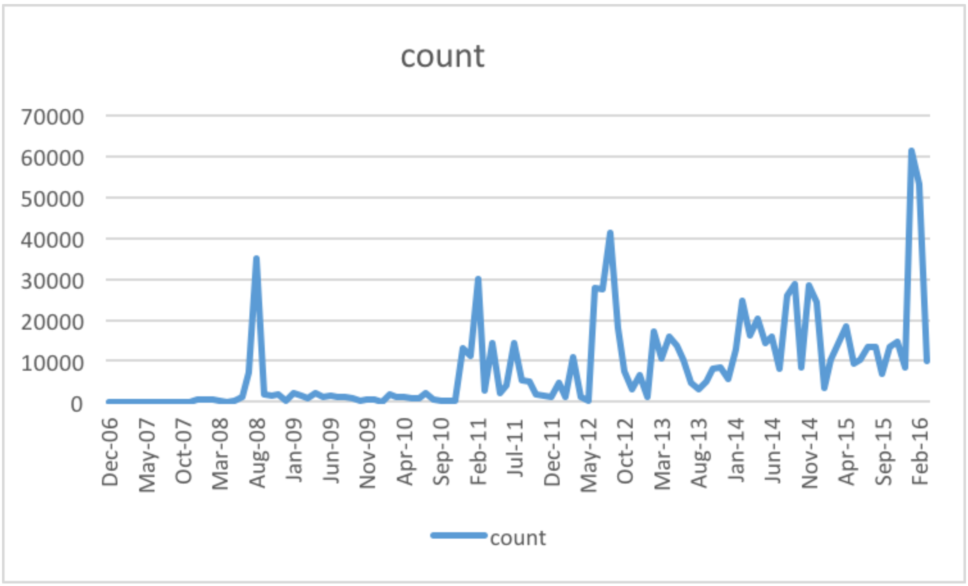
Auditing street names

It would be really nice if there was a automation process which would match similar names even in different languages for each city area and make suggestions for change. This way users will be able to see those similar names and be able to decide which one is correct. Of course there is no guarantee that specific name is the correct one and for this reason it might be based on number of votes. For example, 5 or 10 votes needed for officially selecting name. Same would go for denoting a name. Also a notification might be added if for example a user tries to add name in foreign language. The system might ask if this is not the native name and mark it somehow.

Additional data exploration using MongoDB queries

```
# Activity over time

> db.osm_data.aggregate([
   {"$project": {"month": {"$month": "$timestamp"}, "year": {"$year":
"$timestamp"}}},
   {"$group":{"_id": { "month": "$month", "year": "$year"}, "count":{"$sum":1}}},
   {"$sort": {"_id.year": -1, "_id.month": -1}},
   {"$out": "activity"}])
```



# Longest active users (in days)

```
>db.osm_data.aggregate([
   {"$group": {"_id": "$user.id", "maxt": {"$max": "$timestamp"}, "mint": {"$min":
"$timestamp"}, "entries": {"$sum": 1}}},
   {"$project": {"id": "$id", "time": {"$divide": [{"$subtract": ["$maxt", "$mint"]},
1000*3600*24]}, "entries": "$entries"}},
   {"$sort": {"time": -1}},
   {"$limit": 10}
])

[{"_id": "735", entries": 10, "time": 3198.1935763888887},
   {"_id": "17831", "entries": 1124, "time": 3039.965486111111},
   {"_id": "8421", "entries": 124072, "time": 3003.081585648148},
   {"_id": "38263", entries": 937, "time": 2849.42869212963},
   {"_id": "41176", "entries": 9579, "time": 2810.021469907407},
   {"_id": "44140", "entries": 19936, "time": 2759.0746180555557
   {"_id": "17653", "entries": 188, time": 2725.847511574074},
   {"_id": "31566", "entries": 536, "time": 2649.5729398148146},
   {"_id": "58034", "entries": 261, "time": 2601.2419791666666},
   {"_id": "14610", "entries": 15, "time": 2355.129074074074}]
```

Conclusion

While the size of the data is not bad there is a lot of room for improvement. In the above graphic is visible that user growth has stalled. There should be incentives for popularizing OSM among people - either by gamification or by 3rd party applications. The data import should pass some automated check with suggestions for improvement rather than rejection because users might get upset by rejection but on the contrary might like being suggested with something better.

Sript sources: https://github.com/anenkov/openstreetmap-parser