Project 3: Wrangle OpenStreetMap Data

Introduction

Area Examined: Oslo (county), Norway

In this project I downloaded an XML dataset from OpenStreetMap containing data about Oslo, I converted the data into JSON documents, imported them in MongoDB, run some queries, found inconsistencies and fixed them.

Data Overview

Sizes of XML and JSON files.

```
$ du oslo.osm oslo.osm.json -h
613M    oslo.osm
665M    oslo.osm.json
```

Total number of documents

```
> db.oslo.find().count()
3102937
```

Total number of nodes

```
> db.oslo.find({"__type__":"node"}).count()
2801090
```

Total number of ways

```
> db.oslo.find({"__type__":"way"}).count()
301847
```

Problems Encountered

After modeling the data, uploading it into MongoDB and running some queries I found the following problems:

- some of the postcodes were inconsistent
- many of the postcodes were from other cities
- some of the addresses were incorrect (typos or abbreviations)

Postcodes

According to Wikipedia postal codes in Norway have four digits, start from 0001 and increase with increasing distance from Oslo. The first two digits of the postal code indicate the geographic location the postal code belongs to and in Oslo the first two digits vary from 00 to 12.

About 10.2% of the nodes have a postal code.

```
> db.oslo_data.find({"addr.postcode":{"$exists":1}}).count()
286903
```

The most common postcode is "1450" which is from the Akershus region, a county that borders Oslo (part of the output have been excluded for clarity).

```
> db.oslo_data.aggregate([{"$group":{"_id":"$addr.postcode", "count":{"$sum":1}
{ "_id" : null, "count" : 2816034 }
{ "_id" : "1450", "count" : 4527 }
...
```

Scrolling through the postcodes it can be noticed that a few of them contain the name of the city (for example "1283 Oslo"), a couple of them contain just two digits, and in those cases the first two digits were omitted because they were "00" and there's many of them with the first two digits ranging from "13" to "35"

```
> db.oslo_data.aggregate([{"$match":{"addr.postcode":{"$gte":"13", "$nin":["N-0
{ "_id" : "postcodes", "max" : "3538", "min" : "1311" }
```

All those postcodes come from neighbour counties and the problem turned out to be OpenStreetMap's API that when downloading the data instead of selecting just the data about Oslo it selected all the nodes and ways contained in the rectangular containing the city.

Street Names

In Norwegian most of the time the street type it's appended in the end of the street name, for instance "Aker's street" would be "Akersgata" and this has the advantage that it can't be abbreviated.

However there are some streets that keep the street type separated, and finding them is more complicated than just finding the last word of the street name, because most of the streets don't have this "feature" or sometimes the street type even though it's separated it's not in the end of the street name, so that kind of research would return mostly chunks of street names.

I tried to improve the consistency of the dataset in this aspect writing a regex function to check for abbreviations that I saw on some samples of data and apply the extended version to the whole dataset.

Besides the abbreviations I found that some street names had typos, so I scraped Oslo street names from a local wiki using scrapy,

```
$ scrapy runspider streets_spider.py -o streets.json
```

I imported that data into another collection in MongoDB

```
$ python import.py osm streets streets.json
```

I created a text index

```
> db.streets.ensureIndex({"name":"text"})
```

and then I searched for each street on the map a match on the new collection, calculated the distance between the two strings (using Levenshtein distance) and in case they were close enough (sometimes the match had nothing to do with the actual street name), I updated the street name. It's important to notice that since the majority of the streets were outside Oslo and the majority of the streets were named correctly, a really small part of the street names have been modified.

Further Exploration

Most occurring sources

```
> db.oslo.aggregate([{"$match":{"source":{"$exists":1}}}, {"$group":{"_id":"$sc
{ "_id" : "bing", "count" : 135353 }
{ "_id" : "survey", "count" : 23204 }
{ "_id" : "Bing", "count" : 22698 }
{ "_id" : "PGS(could be inacurately)", "count" : 9185 }
{ "_id" : "Yahoo", "count" : 3093 }
```

Most occurring problem tags

```
> db.oslo.aggregate([{"$match":{"problem_tags":{"$exists":1}}}, {"$unwind":"$pr
{ "_id" : { "class:bicycle:mtb" : "0" }, "count" : 970 }
{ "_id" : { "class:bicycle:mtb" : "1" }, "count" : 887 }
{ "_id" : { "class:bicycle:mtb" : "-1" }, "count" : 224 }
{ "_id" : { "class:bicycle:mtb" : "2" }, "count" : 207 }
{ "_id" : { "class:bicycle:mtb" : "-2" }, "count" : 71 }
...
```

Suggestions

Autocompletion

On the sources list, it can be noticed that Bing appears several times but written in different ways, a way to reduce errors of this kind, on all keys, could be to implement an autocompletion system so that when users insert an existing value they'll have less probability to add it written in a different way. It wouldn't be much difficult to implement, but all the apps and programs that are being used to add data to OpenStreetMaps would have to be updated.

Format Control

Another useful thing would be checking the formats of inserted data, like with postal codes or street names, so that the time spent on cleaning data would be reduced. Even in this case all the apps and programs adding data to OpenStreetMaps would have to be updated.

Conclusion

With this project I realized that whenever there's user input, there's errors and even if there's some standards when inserting data, they will eventually not be followed the whole time or by everyone. By saying this I'm not saying that standards are useless, but that unless they're ensured in a really strict way, they won't guarantee clean data, but just reduce the probability of errors. The data I used seemed well structured and that made the modeling part easier, even if I didn't clean all the fields in the data I found that the cleaning part wasn't the most difficult part of the project, but finding the problems was. Another consideration that I have is that it would be pretty useful if in the OpenStreetMap API it could be possible to download data from just a city instead of delimiting the area with a rectangle.

Sources

- postal codes in Norway
- streets in Oslo
- XML parsing
- read JSON file line by line
- scrapy creating a spider
- strings edit distance