OpenStreetMap Project - San Francisco & Bay Area

David John Wilson - May 2015

Having moved to the Bay Area nine months ago, I have chosen to study the San Francisco area for the OpenStreetMap project. I am using the data set provided by MapZen on their Metro Extracts Page which describes the following OpenStreetMap Area.

The initial OSM file has a size of **648.6MB**.

Cleaning

We start by some basic analysis on the file directly using python. There is a total of 8,278,255 tags. The following are the tag types:

```
{'bounds': 1,
 'member': 42292,
 'nd': 3686723,
 'node': 3010541,
 'osm': 1,
 'relation': 3280,
 'tag': 1211521,
 'way': 323896}
```

Thankfully no tags contained problem characters.

Following the work in Lesson 6, we look at auditing the street names. A surprising amount of normalisation was needed, with the following mapping used:

```
mapping = { "St": "Street","St.": "Street","Ave": "Avenue","Ave.":
"Avenue","Abenue": "Avenue",
"Rd": "Road","Rd.": "Road","Pl" : "Plaza","Pl." : "Plaza","broadway":
"Broadway","street": "Street",
"Plz" : "Plaza","Blvd": "Boulevard","Blvd.": "Boulevard","Boulavard":
"Boulevard","square": "Square",
"parkway": "Parkway","ave": "Avenue","Ln": "Lane","Hwy": "Highway","Dr":
"Drive","Ctr": "Center",
"sutter": "Sutter","Ln.": "Lane","st": "Street"}
```

Other data checks and sanitization required: * **Longitude and Latitude:** ensure they are in the valid range of [[37,38],[-123,-122]]. None were invalid (presumably as MapZen downloads all entries in a certain long/lat range). * **State:** ensure all states are given as "CA". Bad entries included "ca", "California" and "US". * **Country:** ensure all states are given as "US". Bad entries included "us" and "CA". * **Postcode:** ensure all postcodes are given as five digit entries. Bad entries included nine digit postcodes, and postcodes prefixed with CA.

One major issue was that some entries had a tag with key "address". This would overwrite the dictionary that had been created for "address" with a string, and later attempts to add to the 'dictionary' would result in an error.

There was also an issue with processing street names containing the unicode character for a n-tilde (`\xf1`). As the street names get processed by a separate function the string encoding was failing and so the only case where this happened had to be handled as a special case.

After wrangling the data into the suitable JSON format, the produced file has a size of **895MB** (1.4 times the initial OSM file).

Analysis

We now import the data into MongoDB. We use the `pyMongo` library to access the data. We assign `db` to be `db.cities`

Analysis of the Data

We first start by looking at properties of the data itself, such as the number of entries in our database, and some basic features of the data.

We can compute the number of entries in the database with a simple MongoDB command:

```
db.sanfrancisco.find().count()
```

3334437

We can check the number of nodes using MongoDB:

```
db.sanfrancisco.find({"type":"node"}).count()
db.sanfrancisco.find({"type":"way"}).count()
```

3010456 323870

We can compare with the numbers taken from the initial audit of the OSM file and see that 85 nodes and 26 ways are not accounted for. This will be from either empty nodes/ways, or duplication.

We look at the top 5 users:

```
db.sanfrancisco.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                                {"$sort":{"count":-1}},
```

```
                                    {"$limit":5}])
```

ediyes: 729862

Luis36995: 561242

Rub21: 421328

oldtopos: 337543

KindredCoda: 137810

These five users form 65.6% of the total entries.

We look at the top five cities that are actually in the collection:

```
db.sanfrancisco.aggregate([{"$match":{"address.city":{"$exists":1}}},
{"$group":{"_id":"$address.city", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":5}])
```

Redwood City: 23560

Berkeley: 5491

San Francisco: 4531

Piedmont: 3809

Palo Alto: 1655

We see that San Francisco is *not* the top city. In fact, with 43431 entries that have a city field, San Francisco constitutes only a little over 10%. This seems strange for a map area centered on San Francisco. However, as San Francisco extends into the Bay Area (which is also heavily tech-oriented so is perhaps more likely to contain contributors to OpenStreetMap) it can be hard to decide what constitutes a "City" and what is simply an extension of San Francisco.

One reason for the dominance of Redwood City is also that a single user, `oldtopos`, has contributed 23268 entries with this city (98.8% of the entries for Redwood City and 53.6% of all entries with a city). This has clearly created a bias.

For extra clarity we therefore look for all entries with a inner-SF postcode (that start with 941xx):

```
db.sanfrancisco.aggregate([{"$match":{"address.postcode":{"$regex":"^941"}}},
                                    {"$group":{"_id":"dummystring",
```

```
                                    "count":{"$sum":1}}}])
```

2155

There are 8963 entries with a postcode, so 24.0% are in SF. This is larger than the proportion of entries with a city as San Francisco, but is still relatively low for a map centered on San Francisco.

Analysis of the Map Data

We now do some queries related to the map data to see some features about San Francisco and its larger metropolitan area.

What is the most prominent religion?

We look at all amenities that are a `place_of_worship` and group them by religion to see which is the most prominent. Unsurprisingly Christianity is the most prominent, following (a little more surprisingly) by Buddhism. There are 1051 total places of worship with an attached religion (so Christianity constitutes 94.4% of them) and a further 34 without an attached religion.

```
places_of_worship = db.sanfrancisco.aggregate([{"$match":{"amenity":{"$exists":1},
"amenity":"place_of_worship", "religion":{"$exists":1}}},
{"$group":{"_id":"$religion", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":5}])
```

Christian: 992

Buddhist: 25

Jewish: 16

Muslim: 6

Scientologist: 3

What is the most popular type of restaurant?

We look for restaurants that have a `cuisine` entry. The results are unsurprising for San Francisco: Mexican is the most popular cuisine, followed by pizza and Italian. Having the largest Chinatown outside of Asia, it is perhaps surprising that Chinese cuisine is only 4th (although the fact that OpenStreetMap is English-language-focused may have a bias towards English-language cuisines). These results are from 1462 entries with a `cuisine` attached (there are a further 805 restaurants without a `cuisine`).

```
restaurants = db.sanfrancisco.aggregate([
{"$match":{"amenity":{"$exists":1}, "amenity":"restaurant",
"cuisine":{"$exists":1}}},
{"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":5}])
```

Mexican: 184

Pizza: 129

Italian: 107

Chinese: 107

Japanese: 91

What is the most popular fast food restaurant?

Staying on the food theme, we look for fast food outlets (using the amenity type `fast_food`). Although Subway is the most popular fast food chain worldwide, McDonald's is more common in San Francisco. As San Francisco is health conscious it is unsurprising that Jamba Juice features in the top 5 fast food outlets. There are 505 fast food entries in the database.

```
fast_food = db.sanfrancisco.aggregate([{"$match":{"amenity":{"$exists":1},
"amenity":"fast_food"}},
{"$group":{"_id":"$name", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":5}])
```

McDonald's: 46

Subway: 42

Burger King: 28

Taco Bell: 25

Jamba Juice: 16

Are there more drinking fountains or public toilets?

We look at whether public toilets or water fountains are more common. It is unsurprising that water fountains are more common, but it is perhaps surprising that there are only 8 more fountains than toilets.

```
toilets = db.sanfrancisco.aggregate([
{"$match":{"amenity":{"$exists":1}, "amenity":"toilets"}},
{"$group":{"_id":"dummystring", "count":{"$sum":1}}}])

fountains = db.sanfrancisco.aggregate([
{"$match":{"amenity":{"$exists":1}, "amenity":"drinking_water"}},
{"$group":{"_id":"dummystring", "count":{"$sum":1}}}])
```

Number of public toilets: 377

Number of drinking fountains: 385

How many of San Francisco's buildings have wheelchair accessability?

It is vitally important that buildings have sufficient accessability for wheelchairs. Thankfully, the majority of buildings in San Francisco (85.0% to be precise).

```
wheelchairs = db.sanfrancisco.aggregate([{"$match":"wheelchair":{"$exists":1}},
{"$group":{"_id":"$wheelchair", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$limit":3}])
```

Yes: 1039

No: 114

Limited: 70

What is more common: Museums or McDonald's + Starbucks?

A data release from the Institute of Museum and Library Services showed that there are more museums in the United States than McDonald's and Starbucks combined. Unfortunately this does not seem to be true about San Francisco where there are only 96 museums compared to 152 McDonalds and Starbucks. However if you include art galleries and artwork the number increases to 178, which is slightly more heartening.

```
fast_food = db.sanfrancisco.aggregate([
{"$match":{"name":{"$in":['Starbucks',"McDonald's"]}}},
{"$group":{"_id":"dummystring", "count":{"$sum":1}}}])

museums = db.sanfrancisco.aggregate([
{"$match":{"tourism":"museum"}},
{"$group":{"_id":"dummystring", "count":{"$sum":1}}}])
```

Number of Starbucks + McDonald's: 152

Number of Museums: 96

Extensions

I certainly hope to delve further into this data in the future. Of particular interest to me is how the data gets created and how this affects properties of the data. An example above is the fact that a single user created nearly all the entries with the city field Redwood City, and that consisted of half of all city entries - thus skewing the data. This tells us something interesting about the OpenStreetMap data: the fact that one user can skew the city field so much suggests that the city field is not heavily used (or perhaps unintuitive/hidden).

A solution could be to auto-fill this field by longitude/latitude information or infer it when an entry is surrounded by other entries with a city property. There are issues and benefits to both of these approaches.

Auto-filling with respect to longitude and latitude information would require further human input to define the areas of each city (some of which is presumably already in OSM for their search functionality). This may not be accurate, or difficult to decide (for example, in the Bay Area many cities are almost adjoining - I work in East Palo Alto but my office address is for Menlo Park).

Inferring from adjacent entries has the benefit of being programmatic, but has it's own issues. To start, there has to be data already existing for any inference to take place - we have seen with our database that we cannot necessarily make this assumption. Then the rules for inference need to be decided - a first approach may be to infer a city if all entries within a small square surrounding the entry all have the same city. This may work okay for the centre of cities but may run into issues towards the edges of cities (especially when two cities are connected) or cities with unusual shapes (entries in the Vatican City might get misattributed to Rome!). Further, human error can have a large effect on this inference: a single incorrect city entry can prevent inference of all entries near it. Care would also need to be taken that if an entry is surrounded by only a small number of entries then no inference takes place (otherwise if there was only a single entry with a city this would spread like a virus to take over the whole world!).

Perhaps a better solution would be to infer a list of cities using both of the above methods and present the user with a dropdown list of entries (as well as an option to create a new City tag). This helps suggest to the user sensible entries, whilst not totally restricting them. Unfortunately due to the non-homogeneity of naming of cities (much like postcodes, states, and many fields in the OSM data) there are no strict restrictions that can be placed on the entry to prevent dirty data, however warnings can be implemented for cities that match other cities nearby in all but capitalization or hyphenation.

I personally also wish to look more into the entries by the top 5 users and try and identify other biases that these users have (unintentionally) introduced by producing well over half of all OSM data for San Francisco.

Conclusion

Doing this project has taught me a few lessons regarding data wrangling. You certainly cannot expect consistency in data, especially with human generated data. Even fields that would seem to be inherently standardised (such as postcode) need to be looked at carefully. Unfortunately there is nothing that OpenStreetMap can do to combat this - postcodes vary across the world, so putting restrictions on input makes no sense.

I feel like I answered some relatively interesting questions regarding the data. It was also interesting to learn more about the data itself. For example, it was surprising that around one third of the restaurants had no `cuisine` field attached; this sort of fragmentation is another reminder of how difficult working with human-generated data can be.

Anecdotally, it was also very noticeable the difference in the time it took to work with the data when using python directly or MongoDB. The majority of my time was spent in the initial cleaning and wrangling of the data. Once I had produced the JSON data and imported it into MongoDB it was pretty fast to run the queries, even over the 8 million entries.