Udacity Data Analyst Nanodegree

Project Report – Berlin, Germany

Data Wrangling – Case Study: OpenStreetMap Data

Andreas Storz

Choice of metro area

The OSM metro area selected for this project is the city of **Berlin, Germany**. The preselected metro area was obtained via MapZen. I chose this area as I have lived in Berlin for several years and am therefore familiar with the physical location and local specifics (regarding e.g. formatting conventions).

Auditing process & problems encountered

Street names

The auditing process for street names did not reveal any significant problems that should (and could) be addressed programmatically. Unlike U.S. addresses, German street types (e.g., Weg, Allee, Platz) do for the most part not have common abbreviations (such as PKWY, DR, RD). Moreover, abbreviations are overall less common. In addition to that, the street type and actual name are often a single word ("Friedrichstraße"), making it less feasible to split the string and audit the last word of a street entry.

Overall, auditing street name entries showed that virtually no abbreviations were used. Even the fairly common shortening of "Straße" (street) as "Str." did not show up as a problem. Furthermore, "Straße" (by far the most common street type) was also never spelled "Strasse", which is often used to avoid the use of the non-English (and non-ASCII) "ß". Therefore, street names did not pose a problem that required programmatic cleaning.

Phone numbers

Phone numbers showed several inconsistencies in how they were formatted. In order to improve **validity**, phone numbers needed to follow a common standard. Specifically, it was determined they should include the international calling code preceded by one "+" (e.g., +49), followed by the complete phone number, i.e. including the area code in the case of a landline, yet omitting the leading "0". Moreover, phone numbers should not include any whitespace or non-numerical characters such as "/" or "-" (other than the leading "+"). An example of a properly formatted phone number would be: +49302270 (incidentally the phone number of the Bundestag in Berlin). Aside from enhanced validity through standardization, this type of formatting has the added benefit that, when used in web applications, the number can be directly tapped from the screen of a mobile device with a single touch in order to place a call to the given phone. (This method works from any location.)

Examples of improperly formatted phone numbers included altogether missing country codes or poorly formatted ones (“49”, “++49”), as well as redundant characters (“/”, “-“, “()”) and whitespace. Only a very small number of phone numbers could not be standardized programmatically in cases where both country and area codes/prefix were missing entirely (Example: 6767374). In those cases, it could not reliably be judged how to accurately complete these numbers, therefore they were retained in their original form.

Shops vs. amenities

A different issue was detected while auditing shops and amenities, namely inconsistent labeling (as either shop or amenity). In OSM, the value of the field “shop” specifies its type, e.g., “supermarket”. However, its use seemed inconsistent when compared to amenities: While e.g. kiosks, bakeries, and butchers were labeled as “shop”, pharmacies, restaurants, and ice cream parlors would be considered an “amenity”. The judgment ultimately was that all shops are ultimately a form of amenity, and it should be more meaningful to directly know the type of shop a document refers to (such as book shop, liquor store, etc.). Therefore, the category of “shop” was programmatically merged with “amenity”; the value of the amenity field would be the value of “shop”. However, this would introduce vague/uninformative naming given the value of many shops (e.g. “music”, “curtain”). Therefore, all underspecified categories were amended programmatically according to common convention (“massage” -> “massage_studio”); in most cases, the suffix “_shop” was added to the category.

Zip codes

Finally, zip codes were checked for validity. Two types of problems were detected here. First, not all zip codes were consistent with Berlin addresses (Berlin zip codes range from 10115 to 14199). Second, a group of zip codes did not conform to the pattern of (exclusively) 5 digits. It turned out that these documents referred to locations in Poland (e.g., `69–100`). This showed that the Berlin OSM data really refers to a very large metro area which even straddles the Polish border. However, as these entries are not wrong (there is no official definition of the Berlin metro area) but rather unexpected, they were retained in the dataset. Consequently, no cleaning on zip codes had to be performed during the auditing stage.

Overview of the data & exploration

After the auditing/cleaning process was complete, the osm .json file was uploaded into MongoDB (using the following command: `mongoimport –d osm –c osm_berlin ––file berlin_germany.osm.json`).

Size of file

- berlin_germany.osm (XML): 2.72 GB
- berlin_germany.osm.json: 3.02 GB

Total number of documents:

```
> db.osm_berlin.find().count()
13,277,678
```

Number of unique users:

```
> db.osm_berlin.distinct("created.user").length
8,888
```

Number of nodes:

```
> db.osm_berlin.find({"type": "node"}).count()
11,502,951
```

Number of ways:

```
> db.osm_berlin.find({"type": "way"}).count()
1,774,391
```

Top 10 amenities:

```
> db.osm_berlin.aggregate([{"$match": {"amenity": {"$exists": 1}}},
{"$group": {"_id": "$amenity", "count": {"$sum": 1}}}, {"$sort":
{"count": -1}}, {"$limit": 10}])

{ "_id" : "parking", "count" : 14949 }
{ "_id" : "bench", "count" : 12293 }
{ "_id" : "restaurant", "count" : 6011 }
{ "_id" : "post_box", "count" : 3790 }
{ "_id" : "bicycle_parking", "count" : 3642 }
{ "_id" : "recycling", "count" : 3304 }
{ "_id" : "waste_basket", "count" : 3295 }
{ "_id" : "fast_food", "count" : 2921 }
{ "_id" : "kindergarten", "count" : 2809 }
{ "_id" : "cafe", "count" : 2688 }
```

Most popular cuisines for restaurants (top 10) :

```
> db.osm_berlin.aggregate([{"$match": {"amenity": "restaurant",
"cuisine": {"$exists": 1}}}, {"$group": {"_id": "$cuisine", "count":
{"$sum": 1}}}, {"$sort": {"count": -1}}, {"$limit": 10}])

{ "_id" : "italian", "count" : 768 }
{ "_id" : "german", "count" : 565 }
{ "_id" : "regional", "count" : 399 }
{ "_id" : "indian", "count" : 190 }
{ "_id" : "greek", "count" : 182 }
{ "_id" : "asian", "count" : 170 }
{ "_id" : "chinese", "count" : 138 }
{ "_id" : "vietnamese", "count" : 134 }
{ "_id" : "international", "count" : 87 }
{ "_id" : "pizza", "count" : 84 }
```

Other ideas about the dataset

While some programmatic cleaning was performed before the upload, there are certainly many more areas where improvements could be implemented to increase data quality. A few initial ideas are summarized below.

Suburbs

```
> db.osm_berlin.aggregate([{"$match": {"address.city": "Berlin",
"address.suburb": {"$exists": 1}}}, {"$group": {"_id":
"$address.suburb"}}]).itcount()
103
```

As shown above, a query revealed 103 distinct suburbs in documents with addresses and a city name of Berlin. However, officially Berlin is made up of 12 administrative boroughs as of 2001. These boroughs are further subdivided in 96 localities. There are obvious inconsistencies in how the field suburb is coded. Some entries refer to one of the 12 boroughs while others use locality naming. Some may not officially be part of the city state of Berlin at all. First a common standard should be identified, possibly one that uses both borough and locality data (as two fields). Second, existing entries would need to be parsed, possibly also accounting for different ways of spelling and typos. Moreover, where the info is not specific enough to determine a specific neighborhood, position data could be employed. These requirements exceed the resources available for this project.

Website data

```
> db.osm_berlin.aggregate({"$match": {"website": {"$exists":
1}}}).itcount()
20,026
```

As shown above, there are over 20,000 documents that include the field “website”. However, it is not known whether these website addresses are correct and whether they are still valid. Each entry with website data could possibly be checked programmatically to verify that website info is both correct and up-to-date.

Opening hours

```
> db.osm_berlin.aggregate({"$match": {"opening_hours": {"$exists":
1}}}).itcount()
16,707
```

Over 16,000 documents contain info on opening hours of businesses and institutions. However, this info is simply a single string, and there is no convention on consistent formatting, as is illustrated below:

```
> db.osm_berlin.aggregate([{"$skip": 10}, {"$match": {"opening_hours":
{"$exists": 1}}}, {"$project": {"_id": 0, "opening_hours":
"$opening_hours"}}, {"$limit": 20}])
```

```
{ "opening_hours" : "06:00-01:00" }
{ "opening_hours" : "Mo-Su 06:00-18:00" }
{ "opening_hours" : "24/7" }
{ "opening_hours" : "Mo-Fr 07:30-18:30;Sa 08:00-12:00" }
{ "opening_hours" : "Mo-Fr 09:00-18:00; Sa 09:00-13:00" }
{ "opening_hours" : "24/7" }
{ "opening_hours" : "24/7" }
{ "opening_hours" : "Mo-Sa 08:00-22:00" }
{ "opening_hours" : "Mo-Fr 08:00-18:00;Sa 08:00-12:00" }
{ "opening_hours" : "Mo-Sa 09:00-21:00" }
{ "opening_hours" : "Mo-Fr 09:00-18:00; Sa 09:00-13:00" }
{ "opening_hours" : "24/7" }
{ "opening_hours" : "Mo-Sa 16:00-23:00; Su 12:00-22:00" }
{ "opening_hours" : "12:00-24:00" }
{ "opening_hours" : "Mo-Su 09:00-05:00" }
{ "opening_hours" : "Mo-Fr 10:00-18:00; Sa 10:00-15:00" }
{ "opening_hours" : "Mo-Fr 09:00-18:00; Sa 09:30-13:00" }
{ "opening_hours" : "Mar-Oct 09:00-17:00; Nov-Feb 09:00-16:00; Dec 24-
26,31 off; Jan 1 off" }
{ "opening_hours" : "Mo-Fr 09:00-18:30; Sa 10:00-16:00" }

{ "opening_hours" : "Nov-Apr: Mo-Sa 15:00+; Su 12:00+; May-Oct: Mo-Su
12:00+" }
```

Establishing validity by imposing consistent formatting would allow applications to parse the data and use it in applications, e.g., to directly show whether a business is open or closed at the moment. This would however require identifying or building an advanced parser to consistently audit and clean the data. Moreover, a common schema would have to be identified first to capture all relevant information regarding opening times. Finally, ideally all hours would be verified as they may have been entered incorrectly or could have changed in the meantime. However, this would require some form of gold standard data, which is likely hard to come by. Still, at least the format could be harmonized to establish consistency across documents and to allow for automatic parsing of the data and use in applications.

References

N/A