Data Wrangling with MongoDB – OpenStreetMap Data Case Study

Amandeep Mander

Map Area: Waterloo, Ontario, Canada

https://mapzen.com/data/metro-extracts/

Problems Encountered in the Map

After initially downloading the Waterloo area and running it against a provisional data.py file, I noticed the following problems with the data, as discussed below:

- Inconsistent methods of describing street direction (e.g. “N” or “North”)
- Inconsistent street types (e.g. St. instead of Street, Dr instead of Drive)
- Incorrect street types (e.g. AVenue instead of Avenue, Cresent instead of Crescent)
- Inconsistent method of displaying the Postal Code (XXXXXX instead of XXX XXX)

Inconsistent Methods of Describing Street Direction

Some of the street names end with a direction (North, East, South, West). Some of these directions however are entered in as abbreviations (N, E, S, W) as opposed to the full words. In order to keep the direction method consistent, the abbreviation was replaced with the full word.

Inconsistent Street Types

Some of the street types were abbreviated as St. or Dr instead of Street or Drive, respectively. In order to ensure consistency, all the abbreviations were converted to full street types.

Incorrect Street Types

Some of the street types were spelt incorrectly such as AVenue and Cresent instead of Avenue and Crescent, respectively. These incorrect words were detected and replaced with the correct spelling.

Below is the mapping used to correct for the above inconsistencies:

```
mapping = { "St": "Street",
            "St.": "Street",
            "Ave" : "Avenue",
            "Rd." : "Road",
            "Rd" : "Road",
            "AVenue" : "Avenue",
            "Cresent" : "Crescent",
            "Dr" : "Drive",
            "Dr." : "Drive",
            "N" : "North",
            "E" : "East",
            "S" : "South",
            "W" : "West",
            "road" : "Road",
            "g" : "",
            "canadatrust.com" : "",
            "45th" : "45"
            }
```

Inconsistent Method if Displaying Postal Code

In Canada, postal codes are 6 characters long with alternating letters and numbers, with an optional space between the first and last 3 characters (i.e. LNL NLN where L = Letter and N = Number). Some of the postal codes in the dataset have a space between the first and last 3 characters and some do not (e.g. N2K4N2 or N2K 4N2).

In order to make the postal codes consistent, the following code was used to ensure that all postal codes would be the LNL NLN format:

```
if key == 'addr:postcode':
    if not re.match(r'^[A-Z]\d[A-Z] \d[A-Z]\d$', value):
        value = value[0:3] + ' ' + value[3:6]
```

Below is the snippet of code showing that prints out the postal codes in the file, sorted by descending count:

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

As can be seen from the output below, the data going into the database contains postal codes in the LNL NLN format:

```
{u'_id': u'N2K 4L7', u'count': 70}
{u'_id': u'N2K 4N3', u'count': 66}
{u'_id': u'N2K 4N2', u'count': 50}
{u'_id': u'N2B 1A4', u'count': 42}
{u'_id': u'N2M 3V1', u'count': 38}
{u'_id': u'N1G 4C9', u'count': 30}
{u'_id': u'N2T 0A6', u'count': 28}
{u'_id': u'N2J 1K7', u'count': 26}
{u'_id': u'N2K 4M3', u'count': 22}
{u'_id': u'N2B 1G9', u'count': 20}
{u'_id': u'N2E 0A3', u'count': 18}
```

Data Overview

Below are some of the basic statistics about the dataset, including the MongoDB queries used to produce the statistics.

File Size

waterloo-region_canada.osm – 155MB

waterloo-region_canada.json – 211MB

Number of Documents

db.waterloo.find().count()

→ 763051

Number of Nodes

db.waterloo.find({"type" : "node"}).count()

→ 670362

Number of Ways

db.waterloo.find({"type" : "way"}).count()

→ 92689

Number of Unique Users

db.waterloo.aggregate([{ "$group": { "_id": "$created.user"}},

{"$group": {"_id": 1, "count": { "$sum": 1}}}])

→ 503

The osm file did not contain any information about amenities.

Additional Ideas

Below is a snippet of code that prints out the city names:

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

Below is a sample of the produced output:

```
{u'_id': u'City of Guelph', u'count': 10128}
{u'_id': u'City of Cambridge', u'count': 3597}
{u'_id': u'Township of Woolwich', u'count': 2789}
{u'_id': u'Township of North Dumfries', u'count': 1491}
{u'_id': u'Township of Centre Wellington', u'count': 1491}
{u'_id': u'Township of Wellesley', u'count': 1418}
{u'_id': u'Kitchener', u'count': 717}
{u'_id': u'Township of Guelph/Eramosa', u'count': 608}
{u'_id': u'Township of Perth East', u'count': 571}
{u'_id': u'Township of Mapleton', u'count': 556}
{u'_id': u'Waterloo', u'count': 444}
{u'_id': u'City of Waterloo', u'count': 306}
```

As can be seen, some of the city names are duplicated with different names, such as Waterloo and City of Waterloo. One way to fix this issue would be to either include the type of city (e.g. City of, Township, etc.) for all cities or not include it for any of the cities. The issue with this approach is that the title is part of the official name for some of the cities and not for others, so it would inappropriate to force the

names to be consistent in this case. The benefit of sticking to one method is that there will not be any duplicates, but forcing consistency may result in erroneous city names.

Conclusion

Although some of the inconsistent and incorrect data were fixed, there still remain some inconsistencies and errors. The street directions and types were mostly corrected for whereas the postal codes and city names remain inconsistent. Altering the postal codes to be more consistent is fairly straightforward whereas correcting the city names would pose a challenge.