

Wrangling OpenStreetMap Data

Introduction

This project was done as part of the Data Analyst Nanodegree program. The goal is to take raw data of a chosen geographical area from the OpenStreetMap project, audit the data, fix the most occurring and identify any other problems, import the data into a SQL database and explore it. I have chosen Prague, Czech Republic and its surrounding area as I have lived there for five years and I am therefore more interested in exploring the data.

Data source

The full dataset containing all of Prague, a large part of Central Bohemian region and a small part of Usti nad Labem region was available on July 8, 2017 for download here:

- https://mapzen.com/data/metro-extracts/metro/prague_czech-republic/
(https://mapzen.com/data/metro-extracts/metro/prague_czech-republic/)

Problems Encountered in the Map

The whole dataset has almost 2GB uncompressed so I have started by subsampling using the provided script which takes every k-th element. I have chosen k to be 500 and the resulting file was around 4MB. This smaller dataset was useful to find the most obvious issues such as use of non-ascii characters, different postcode formats and abbreviations in street names. It has also served well to write basic consistency check function for house identification numbers but many issues with these numbers appeared only after auditing the whole dataset.

Local characters

The code has to assume utf8 since Czech language has non-ascii characters.

Postcode

Postcode is composed of five digits which are sometimes written without spaces and sometimes with a single space between third and fourth numbers. See an example below:

```
<tag k="addr:postcode" v="13000" />
<tag k="addr:postcode" v="130 00" />
```

I convert the latter representation into the former. Additionally, I have found some individual problems with postcode which would have to be addressed manually. There are postcodes which include also city name and district or they have different number of digits, for example.

Unfortunately, there was no pattern connecting these individual problems.

Street name

Streets in Czech Republic don't have the word for street (Czech: "ulice") in their name. There are very few exceptions such as avenue (Czech: "třída") and square (Czech: "náměstí"). I have discovered that some street names in the dataset include abbreviated 'nám.' in their name. I expand all 'nám.' occurrences in street names into 'náměstí'.

House identification numbers

The most challenging part of this project was verification of consistency of house identification numbers. Let me first introduce the house numbering system in use in Czech Republic. First identification numbers were introduced in 1770 when the country was part of Austria-Hungarian Empire. These are called conscription numbers (**cID**, Czech: "popisné číslo"). Even though, some cities changed cIDs during the centuries, the property of cID stayed. Every house has to have a cID and it has to be unique to a city. There are exceptions for temporary and recreational buildings. Those have so called provisional numbers (**pID**, Czech: "evidenční číslo") and do not have cIDs. Similarly to cID, every pID has to be unique to a city. Nevertheless, note that a city can have a building which has a cID identical to some other building's pID.

To make things easier (or complicated, depending on the point of view), the emperor declared in 1857 that new numbers, also called orientation numbers, will be used in the large cities such as Prague. In OpenStreetMap, these are named street numbers (**sID**, Czech: "orientační čísla"). Nowadays, sID is an optional identification number of a building and it is unique to a street. The idea is that sIDs in a street are ordered which makes it easier to find a building you are looking for. If a new building is build between two existing ones, a letter is used as a postfix of sID. Hence, sIDs can have a single letter at the end.

All the identifiers are composed into house numbers (**hID**) in OpenStreetMap. The format of hID should be

- **cID** if only cID is known
- **ev.pID** if only pID is known
- **cID/sID** if cID and sID are known
- **ev.pID/sID** if pID and sID are known

See an example node:

```
<node changeset="25412874" id="296444099" lat="50.0820084" lon="14.4564234" timestamp="2014-09-13T14:59:25Z" uid="411213" user="pedro" version="3">
  <tag k="addr:street" v="Táboritská" />
  <tag k="addr:postcode" v="13000" />
  <tag k="addr:housenumber" v="15/22" />
  <tag k="addr:streetnumber" v="22" />
  <tag k="addr:conscriptionnumber" v="15" />
</node>
```

These are the problems and solutions relevant to the IDs:

- Missing IDs and we know that they should be there
 - hID missing and other IDs known and consistent
 - Compose hID from others
 - hID known and some other IDs missing
 - Parse hID and create others
 - Only sID known
 - Compose hID as '/sID'
- IDs are in incorrect form
 - hID is one of the following ['?/sID', '/sID', 'sID'] and sID is known
 - Remove hID before standard processing
 - Only hID is known and it is a single number with a letter at the end
 - Assume that it is a sID and both cID and pID are unknown
 - pID has an extra space in hID after 'ev.'
 - Remove the space


There are other issues which I did not tackle as they cannot be solved programmatically or do not occur frequently. I do detect them so that they could be inspected during audit but I keep the tags as they are during processing. These issues include:

- Inconsistencies
 - cID, pID or sID do not correspond to hID
- Multiple identifiers of the same type
 - A house with two cIDs, two pIDs or two sIDs might exist as a result of an administrative mistake
- Different formats
 - hID is sID/cID instead of cID/sID
 - Various other non-standard formats

Please refer to the code for implementation of the consistency check function.

Overview of the Data

Please see schema.sql for details on the database schema. We have used the suggested schema with one modification. Every node and way has a user id (field: uid) and username (field: user). We keep only user id and create a new table for users

```
 CREATE TABLE users (  
    id INTEGER PRIMARY KEY NOT NULL ,  
    username TEXT  
);
```

File sizes

```
prague.osm ..... 1.82 GB
prague.db ..... 0.97 GB
nodes.csv ..... 583 MB
nodes_tags.csv ..... 116 MB
ways.csv ..... 46 MB
ways_tags.csv ..... 121 MB
ways_nodes.csv ..... 233 MB
users.csv ..... 53 kB
```

Numbers of records in tables

```
SELECT COUNT(*) FROM table_name;
```

where table_name is one of the following:

```
nodes          8 479 625
nodes_tags     3 477 775
ways           969 358
ways_nodes    10 222 727
ways_tags      3 711 139
users          2 954
```

Amenities

Lets have a look what are the most common amenities

```
SELECT value, COUNT(*) num FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

```
bench          3 962
recycling       2 585
restaurant     2 563
bicycle_parking 986
post_box       844
telephone      755
shelter        745
pub            711
cafe           660
atm            602
```

Since we have data from Czech Republic, which is know for its beer, we could ask for all the pubs and restaurants which make their own beer with the following query

```
SELECT name.value, street.value, housenumber.value, city.value
```

```

FROM (SELECT DISTINCT(id) FROM nodes_tags
      WHERE key='amenity' AND
            (value='pub' OR value='restaurant'))
      ) AS pubs
JOIN (SELECT id FROM nodes_tags
      WHERE key='microbrewery' AND value='yes')
      ) AS own_beer
ON pubs.id=own_beer.id
JOIN (SELECT id, value FROM nodes_tags
      WHERE key='name')
      ) AS name
ON pubs.id=name.id
JOIN (SELECT id, value FROM nodes_tags
      WHERE key='street')
      ) AS street
ON pubs.id=street.id
JOIN (SELECT id, value FROM nodes_tags
      WHERE key='city')
      ) AS city
ON pubs.id=city.id
JOIN (SELECT id, value FROM nodes_tags
      WHERE key='housenumber')
      ) AS housenumber
ON pubs.id=housenumber.id;

```

```

Pivovarský dům, Ječná, 16, Praha
U Fleků, Křemencova, 1651/11, Praha
Hostinec nad Šárkou, Evropská, 209, Praha
Hospudka U Mlýna, U Mlýna, 8, Zadní Třeban
U Medvídků, Na Perštýně, 7, Praha
Novoměstský pivovar, Vodičkova, 20, Praha
Kláštevní pivovar Strahov, Strahovské nádvoří, 10, Praha
Berounský medvěd, Tyršova, 135, Beroun
Pivovar MMX, Pražská, 452, Lety
U Valšů, Betlémská, 5, Praha
U tří růží, Husova, 232/10, Praha
Pivovarská restaurace, Rýznerova, 19/5, Únětice
Pivovar Victor, Husitská, 72/35, Praha
Pivovar Hostomice pod Brdy, Pivovarská, 214, Hostomice

```

Additional Ideas

I believe that the data is in a good shape but there is still some space for improvement. In order to improve quality, I think that the data should be unified. Consider the following queries before discussing why.

Lets investigate how many nodes and ways did the top users make.

```

SELECT username, count(username) AS num
FROM users
LEFT JOIN (SELECT nodes.uid FROM nodes

```

```

        UNION ALL
        SELECT ways.uid FROM ways) as tags
    ON users.id=tags.uid
GROUP BY users.id
ORDER BY num DESC
LIMIT 10;

```

```

JandaM          2 103 155
Petr1868        1 897 817
pedro'          406 657
pschonmann      387 865
kwiecpav        358 751
CzechAddress    348 128
Minimalis       324 936
BiIbo           248 674
montardo        223 648
Jindřich Houska 218 961

```

We can also look at the proportion of all nodes and ways that the top users made by reusing the previous query

```

WITH counts AS (
    SELECT username, count(username) AS num
    FROM users
    LEFT JOIN (SELECT nodes.uid FROM nodes
        UNION ALL
        SELECT ways.uid FROM ways) as tags
    ON users.id=tags.uid
    GROUP BY users.id
)
SELECT username, ROUND(CAST(num AS FLOAT) / (SELECT SUM(num) FROM counts), 3)
    FROM counts
    ORDER BY num DESC
    LIMIT 10;

```

```

JandaM          0.223
Petr1868        0.201
pedro'          0.043
pschonmann      0.041
kwiecpav        0.038
CzechAddress    0.037
Minimalis       0.034
BiIbo           0.026
montardo        0.024
Jindřich Houska 0.023

```

Interestingly enough, it seems that there is no single user responsible for most of the data. This means that there was no single huge data transfer from another source when creating the map data. Since the data was created by different users, it might have been inserted using different patterns. See the following query for further prove that the data comes from various sources

```

SELECT value, COUNT(*) AS num
  FROM nodes_tags
 WHERE key='source' OR key='created_by'
 GROUP BY value
 ORDER BY num DESC
 LIMIT 10;

```

```

JOSM                41 478
Merkaartor 0.13      13 494
http://www.bnhelp.cz" 7 188
Merkaartor 0.12      5 242
bing:ortofoto        4 563
cuzk:km              4 169
survey              3 531
shpupload            3 124
mvcr:adresa         1 863
Bing;cuzk            1 708

```

My suggestion is to try to cluster the data based on the way they were created. This might be very difficult but I would start by clustering based on a user and year an element was created. Then I would consolidate the data so that they are saved using the same process. This would could solve some ambiguities in the house indentification number verification for example.

Conclusion

In this project, I took map data of Prague, Czech Republic and its surrounding areas from OpenStreetMap project. I have implemented scripts for auditing the most important fields and for fixing the most occuring errors. I have also identified some other errors which would have to be manually fixed. I have used the auditing scripts for cleaning the data and saved the data into .csv files. Furthermore, I have then created an SQL database from the .csv files and explored the data. I have also proposed how can the data be further improved.

Additionally, I have finally understood what do the orientation numbers in Czech cities mean.

Files

The whole project was implemented in jupyter notebooks in python and the functionality was then split into individual .py files as instructed. Therefore, the .py scripts have the same functionality as the wrangle-prague.ipynb notebook. Use audit.py for running audit and clean_and_convert.py for cleaning and parsing into csv. Note that Anaconda software was used for managing python and its packages. The used environment is available in environment.yaml.

Additional Resources

utf8 pprint function taken from <https://stackoverflow.com/questions/10883399/unable-to-encode-decode-pprint-output> (<https://stackoverflow.com/questions/10883399/unable-to-encode-decode-pprint-output>)

pprint-output)