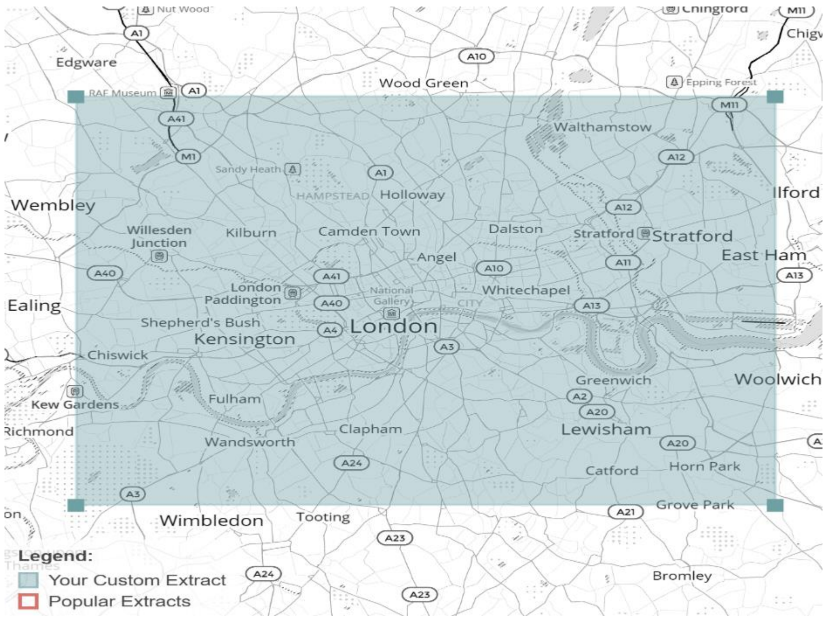
OpenStreetMap Data Case Study

Map Area

London, England, United Kingdom
https://www.openstreetmap.org/relation/65606
https://mapzen.com/data/metro-extracts/metro/london_england/

London is the capital of England and the United Kingdom. It attracts tourists worldwide for its history, culture, entertainment. As a football fan, I like the city for its famous football clubs, like Arsenal, Chelsea, Hotspur. So I will explore this area and see what I can get from the dataset. The entire London has grown to be very large, in this study, I'll only concentrate on the central area of the city, as shown in the selected area.



Problems encountered in the map

After downloading the raw dataset of the area, samples and the whole data set are examined. The following problems are found:

Street names

- Abbreviated street names like "Camden High St"
- Both capital and lowercase names like "Loman street"
- Spelling mistake like "Roger Steet"

Phone number

- Not in consistent format like "+44 207 436 7739" "+44 20 76519271" "+44 02078280235"

Clear Street names

The most common problem is the inconsistency in the format of street names. Some street names use abbreviated names like 'St.','N', while some use the full name 'Street', 'North'. It's also found that some cases use lowercase names for the first character like "street", and some have spelling mistakes like 'Steet'. All these cases, the street name will be modified to full name with the first character in capital. In the 'audit.py' file, the varaible 'mapping' reflects the changes needed to fix the unexpected street types to the appropriate ones. And the names are updated by the following function:

```
def update_name(name, mapping):
        parts = []
    # Split the name and check each part against the keys in the mapping
    # If exists in the mapping key, overwrite that part with the corrected value.
        for name_split in name.split(" "):
                if name_split in mapping.keys():
                        name_split = mapping[name_split]
                parts.append(name_split)
        name_updated = " ".join(parts)
        return name_updated
```

Examples of updates by the function:
- "450 Kingsland Rd" -> "450 Kingsland Road"
- "Essex road" -> "Essex Road"
- "Roger Steet" -> "Roger Street"

Clear phone numbers

The format of contact phone number is another problem with the dataset. The phone number consists of the country code(+44), area code(20, mobile phone will differ), and 8 digits. The phone number in the dataset has different empty space positions, like "+44 207 436 7739", "+44 20 76519271", "+44 20 7241 3377". Some of the numbers have extra 0 before the area code, like "+44 02078280235". Some of the numbers use '-' instead of empty space, like "+44-207-630-1000". Some of the numbers have no country code, like "020 7487 5077". All of these phone numbers will be adjusted to the same format of "+44 XX XXXXXXXX".
The phone numbers are checked in 'audit.py' using the following function:

```
import re
phone_format    = re.compile(r'^\+44\s\d{2}\s\d{8}$') #unifed format
def update_phone(phone):
        m =    phone_format.search(phone)
        if not m:
                bits = filter(str.isdigit, phone)
                if(len(bits)==12):                    # if contains 12 digits, arrange the format
                        phone_updated = "+"+bits[0:2]+" "+bits[2:4]+" "+bits[4:12]
```

```
            elif(len(bits)==13 and bits[2]=='0'):           # extra 0 before area code
                phone_updated = "+"+bits[0:2]+" "+bits[3:5]+" "+bits[5:13]
            elif(len(bits)==11):
                phone_updated = "+44"+" "+bits[1:3]+" "+bits[3:11]
            else:
                phone_updated = phone
        else:
            phone_updated = phone
        return phone_updated
```

Examples of updates by the function:
- "+44 207 436 7739" -> "+44 20 74367739"
- "+44 02078280235" -> "+44 20 78280235"
- "+44-207-630-1000-" -> "+44 20 76301000"
- "(020) 7487 5077" -> "+44 20 74875077"

Prepare for the database

After auditing and cleaning the data, the elements in the XML files are transformed to tabular format, and write to .csv files, using the 'data.py' script. Five csv files are generated, "nodes.csv", "nodes_tags.csv", "ways.csv", "ways_nodes.csv", "ways_tags.csv". The schema can be checked in the 'schema.py' file. Then, these csv files are imported to a SQL database as tables, using the “csv-sql.py”.

Data Overview

This section contains some basic statistics about the dataset, and the SQL queries used to gather them.

File sizes

- London.osm 262 MB
- London.db 155 MB
- nodes.csv 85.6 MB
- nodes_tags.csv 14.3 MB
- ways.csv 12.1 MB
- ways_tags.csv 23.4 MB
- ways_nodes.csv 35.3 MB

Number of nodes

```
sqlite> .open London.db
sqlite> SELECT COUNT(*) FROM nodes;
1076818
```

Number of ways

```
sqlite> SELECT COUNT(*) FROM ways;
210222
```

Number of unique users

```
sqlite> SELECT COUNT(DISTINCT(e.uid)) FROM (SELECT uid FROM nodes UNION
SELECT uid FROM ways) as e;
2808
```

Restaurants in the selected area

```
sqlite>   SELECT count(*) FROM nodes_tags as b WHERE b.value='restaurant';
1896
```

The top 5 popular cuisine

```
sqlite> SELECT b.value, count(*) as num FROM nodes_tags as b, nodes_tags as c
            WHERE b.id=c.id
            AND c.value='restaurant'
            AND b.key= 'cuisine'
            GROUP BY b.value
            ORDER BY num DESC
            limit 5;
italian|170
indian|113
japanese|79
chinese|68
pizza|53
```

The result is interesting that restaurants with most numbers are foreign styles, like Italian or Asian, which indicate they are more popular in the area.

Conclusion and additional ideas

In this project, the dataset of the selected area of London from OpenMapStreet has been audited, cleaned, and imported into SQL database. And one can search information of interest from the SQL database. It is quite convenient for providing many information. In the process of auditing the data, it’s seen that most of the items in the original dataset are in good format. But there are also some items with missing part of the values, or not in consistent format with other items. This is often the case with user input data.

To improve the data quality, OpenStreetMap should use more strict user input data format check and remind user of possible error when the format doesn’t match. The benefit is obvious in reducing the work needed in cleaning. But the problem would be users may find discouraged in dealing with the format. A compromise proposal is that OpenStreetMap can provide users with nearby examples for each entry. Users will make less mistakes with references. And OpenStreetMap can use a format check tool that can auto change the input data for common problems like mentioned in this project, and remind user if the input doesn’t match the unified format or the check tool doesn’t know how to deal with it.

Reference

https://s3.cn-north-1.amazonaws.com.cn/static-documents/nd002/sample_project_en.pdf
https://s3.cn-north-1.amazonaws.com.cn/static-documents/nd002/SampleDataWranglingProject_en.pdf
https://docs.python.org/2/library/xml.etree.elementtree.html#module-xml.etree.ElementTree
http://stackoverflow.com/questions/2887878/importing-a-csv-file-into-a-sqlite3-database-table-using-python
http://stackoverflow.com/questions/3425320/sqlite3-programmingerror-you-must-not-use-8-bit-bytestrings-unless-you-use-a-te