Toggle code

Project OpenStreetMap Data Wrangling

The goal of this project is to learn how to wrangle data and assessing the quality of the data for completeness, uniformity, validity, accuracy, consistency.

Objectives ¶

• Assess the quality of the data for validity, accuracy, completeness, consistency and uniformity.

• Parsing and gather data from popular file formats such as .json, .xml, .csv, .html.

• Process data from many files and very large files that can be cleaned with spreadsheet programs.

• Learn how to store, query, and aggregate data using SQL.

Project Introduction

The objectives of this project are to gather data set from OpenStreetMap, audit and clean it, convert the data from XML to CSV, and insert it to MySQL database. The area analysed in this project will be Dublin, Ireland (https://www.openstreetmap.org/relation/1109531 (https://www.openstreetmap.org/relation/1109531)) Since I moved to Dublin recently I will analyse this area.

The Dataset

OpenStreetMap data is structured in XML document (.osm file) that consist of the following elements:

• Nodes (defining points in space),

• Ways (defining linear features and area boundaries), and

• Relations (which are sometimes used to explain how other elements work together).

Auditing the Tag Types

In order to audit the data each type of existing tags had to be found. “Auditing the Tag Types.py” code was used to iteratively parse through the file and to return tag type as well as the number for each of them.

{'bounds': 1, 'member': 84777, 'nd': 1835235, 'node': 1342665, 'osm': 1, 'relation': 4782, 'tag': 999629, 'way': 243643}

Auditing the k Tags

'k' tags were audit. Three different regular expressions to filter these tags were used to check for any problems before importing the data into SQL database.

The first regular expressions checks for tags with only lowercase letters, the second regular expressions or lowercase letters separated by a colon, and the third regular expressions flags any unwanted characters.

iterparse method of ElementTree was used to compile a list of tags that fell into one of the three regular expression listed above. Auditing the k Tags.py code was used.

```
{'lower': 626160, 'lower_colon': 330960, 'other': 42509, 'problemchars':
 0}
```

{'lower': 626160, 'lower_colon': 330960, 'other': 42509, 'problemchars': 0}

Auditing the Users

Auditing the Users.py have been used to find out how many different users have contributed to the selected data set.

```
1523
```

1523

Subsetting the Data

Due to the size of the data set, Subsetting the Data.py code was used for subsetting so it could be faster to find possible problems and run test code to fix those problems. k = 1000 was used for first subsetting and then k was reduced in order to include more data. At the end, original size data was used.

Data Assessment

Some of the problems of the dataset which were found:

• Abbreviations of street types like ‘Rd’ instead of ‘Road’

• All lowercase letters like ‘road instead of ‘Road.

• Inconsistent postal codes. Users were adding many different types of codes. Three different codes will be acceptable for this project (example of acceptable: Dublin 1, D01 and D05 N7F2).

• The 'k' tags did not follow the specific format.

Auditing Street Names

The audit of the street names should extract non acceptable ones form the XML. This process was iterative (more and more were added to mapping). At the end, list was cleaned manually since there is lots of singles. In Ireland it is common that street names don't have suffix such as Street. For auditing Audit street.py code was used.

Improving Street Names

To clean the abbreviated street names ImprovingStreetNames.py was used. Example: Ave to Avenue or Rd to Road or lane to Lane.

Auditing Postcode

Inconsistent postal codes. Users were adding many different types of codes. Three different codes will be acceptable for this project (example of acceptable: Dublin 1, D01 and D05 N7F2). There will be lots of singles in this cleaning. Reason for it is irish Eircode system in Ireland. Each unit (appartment, house, pub, etc.) have onw number such as D6W XK28. For auditing Auditing Postcode.py code was used. Reason for accepting three different version is they are commonly used in Ireland.

Improving Postcode

Below is listed what will be imporved. Postcode without D will be improved to state D04 for example. And two Eircodes will be created as D05 N7F2.

{'12': set(['12']), '13': set(['13']), '17': set(['17']), '2': set(['2']), '22': set(['22']), '3': set(['3']), '4': set(['4']), '8': set(['8']), 'D05N7F2': set(['D05N7F2']), 'D6WXK28': set(['D6WXK28'])}

Also postcodes with only zeros will be removed.

Preparing the Data for the SQL database

To prepare the data for the SQL database, it is parsed through the XML data, all the improvements are implemented and converted it into tabular form. At the end csv files are created.

The code for this process is Data for SQL.py.

Used schema.py

Analysing of the Data

Files size:

dublin_ireland.osm 315.369KB

nodes.csv 107.687KB

nodes_tags.csv 5.556KB

ways.csv 17.179KB

ways_nodes.csv 44.184KB

ways_tags.csv 28.492KB

Creation of the Tables

Code below was used to create the Tables which were later analysed. Code was obtained from https://gist.github.com/swwelch/f1144229848b407e0a5d13fcb7fbbd6f (https://gist.github.com/swwelch/f1144229848b407e0a5d13fcb7fbbd6f)

Statistical overview of the Data

Next SQL queries were used to obtain statistical information from the data.

Number of ways:

243644

Number of nodes:

1342666

Number of unique users:

1498

Top 10 zipcodes:

D06 691 D6W 567 D07 218 D08 218 D02 100 D04 92 D03 89 D12 81 D14 70 D22 50

Number of cafes, hotels, pubs, and restaurants:

Cafe 684 Hotel 243 Pub 714 Restaurant 768

Cuisine

Coffee shop 123 Chinese 108 Italian 101 Fish and chips 90 Burger 84 Pizza 77 Indian 60 Sandwich 47 Asia 39 Thai 39

Religion

Christian 492 Multifaith 11 Jewish 6 Buddhist 4 Muslim 2 Bahai 1 Hindu 1 none 1 Sikh 1

Other ideas about the dataset

Improvement of the Dataset

• The data could be improved by standardizing the information imported. For example some tags provide excellent information about location while some none or very limited details.

• Standardization of input for postcode in Ireland will be of great help. Postcode should be limited only to Eircode as option. Great use of GPS for example. Lots of addresses are hard to find and this option in my opinion is great benefit.

• Data inserted by users shows enormous inconsistencies. “Forcing” the users to insert minimal requirements might be good but also bed idea.

Potential problems

• “Standardization of input for postcode in Ireland will be of great help. Postcode should be limited only to Eircode as option.” This might not be best option for people living outside Ireland (I am pretty sure that this will be confusing for them). Also when importing data, maybe, if automated script can use address provide as an input to https://finder.eircode.ie/#/ (https://finder.eircode.ie/#/) and return eircode and automatically fulfil postcode tag.

• “Forcing the users to insert minimal requirements might be good but also bed idea.” Some of the users might think that this is too much and could possibly give up. But certain minimum should be set.

• Adding wrong data and how to check if something is wrong or not. There should be a script like/dislike/wrong/helpful or script to report wrong data with possible check for all inputs made by specific user to check if he/she is deliberately entering wrong data.

References

https://www.openstreetmap.org (https://www.openstreetmap.org)

http://wiki.openstreetmap.org/wiki/Elements (http://wiki.openstreetmap.org/wiki/Elements)

https://finder.eircode.ie/#/ (https://finder.eircode.ie/#/)

https://docs.python.org/2.7/ (https://docs.python.org/2.7/)

https://stackoverflow.com (https://stackoverflow.com)

https://gist.github.com/swwelch/f1144229848b407e0a5d13fcb7fbbd6f
(https://gist.github.com/swwelch/f1144229848b407e0a5d13fcb7fbbd6f)

https://stackoverflow.com/questions/27934885/how-to-hide-code-from-cells-in-ipython-notebook-visualized-with-nbviewer (https://stackoverflow.com/questions/27934885/how-to-hide-code-from-cells-in-ipython-notebook-visualized-with-nbviewer)