OSM Data-Wrangling

OpenStreetMap Data Case Study

Map Area

Chicago, IL United States

Chicago is a beautiful and very diverse city. It is one of my favourite place. So, I decided to pick City of Chicago for this exploratory data analysis. I downloaded the Chicago metro area from Mapzen. https://mapzen.com/data/metro-extracts/your-extracts/f5b03d19a3d2 (https://mapzen.com/data/metro-extracts/your-extracts/f5b03d19a3d2)

Introduction

Open Street Map (OSM) is a collaborative project that creates and distributes free geographical information accessible to everyone around the world. In this project, OSM data was assessed for its validity, accuracy, consistency and completeness. The cleansed data was imported into SQlite database, so that it could be queried. First iterative parsing was done which iterates through each element of the .osm file. This step tells us about the tags and their numbers in the dataset. The root elements (node, ways) and child elements (tag, nd) were extracted. Nodes are specific points on earth surface defined by longitude and latitude. Ways define linear features and area boundaries like Street, Avenue, Drive, Road etc.

First I downloaded the preselected map area from Mapzen in XML format (chicago_il.osm). Since this original file was quite big of size 1.56GB, I used the code provided in the instructor note and created a smaller sample.osm file (158MB). Output of this function is a sample.osm file, which is derived from the original .osm file and will reduce the run time. The size of sample file can vary on the basis of the value of k.

```
OSM_FILE = "chicago_il.osm"  #osm file
SAMPLE_FILE = "sample.osm"
k = 10 # Parameter: take every k-th top level element
```

After the sample file was created, function for iterative parsing was used to process the dataset and see the tag name and number for parent element and child element.

```
{'bounds': 1,
 'member': 68709,
 'nd': 7492487,
 'node': 6304534,
 'osm': 1,
 'relation': 2535,
 'tag': 5511964,
 'way': 996230}
```

Before processing and adding the data into the database, data was checked for "k" value for each "" and see if there were any potential problems.

Tags were categorized into four groups as follows:

1. "lower", for tags that contain only lowercase letters and are valid,
2. "lower_colon", for otherwise valid tags with a colon in their names,
3. "problemchars", for tags with problematic characters, and
4. "other", for other tags that do not fall into the other three categories.

{'lower': 1456893, 'lower_colon': 2541289, 'other': 1513779, 'problemchars': 3}

Problems Encountered in the Map

After auditing data for uniformity, validity and consistency, several problems were noticed.

1. Inconsistent, abbreviated and Misspelled streetnames (Washington St, Oakton St., E North ave, North Milwaukee Avebue)
2. Inconsistent format of Postcodes ('IL 60605-1226','60654-5799', '46320')
3. Invalid and incorrect postal codes ('2300', '606', '46320','46327','46321', '46394' -- Some invalid numbers were recorded as zip codes. Some zip codes which did not belong to chicago metro area were also noted in the map)

Inconsistent abbreviation of Street Names

Very inconsistent pattern of abbreviation was observed among the street names during data audit. This abbreviation was not consistent and sometimes street names included spelling errors too. To address this problem,"update_name" function was used which was a part of exercise in the course. This function involved a dictionary "mapping" which included pair of abbreviated names to unabbreviated ones. Each word in address was iterated and replaced with the correct one as stored in mappings using a update_name function (4audit.py).

```
def update_name(name, mapping):
    m = street_type_re.search(name)
    if m.group() not in expected:
        if m.group() in mapping.keys():
            name = re.sub(m.group(), mapping[m.group()], name)
    return name
```

This function updated all the abbreviated, inconsistent and incorrect substrings.

Postcodes

In US, the standard pattern of format consists of five digits. An extended 'ZIP+4' code, includes the five digits of the ZIP Code, a hyphen, and four additional digits which provide more specific location information within a given ZIP Code. Another category of values those needed cleaning in our database were post codes. On our audit,very inconsistent pattern of postcodes distribution were noted. Besides the set of standard Zip codes (five digits), we noticed --Several zip codes with leading characters ('IL, 60642','IL, 60126',) were found. --Several zip codes with 9 digit format ('60660-1607','60160-1607') were found. --Some 4 or 3 digits number values were present as zipcodes. For example- ('6074','6017','6017', '606'). Since Chicago metro area zip codes begin with 60--- it seemed like these numbers were missing last digits. However, this did not hold true for some 4 digit numbers like 0201','2300'. --Several zipcodes ('46394', '46320', '46327') were found which did not belong to Chicago area rather they were from adjacent cities Whiting,IN or Hammond IN. During auditing of the postcodes, a regular expression (r'^6\d\d\d\d$' -- must have five digits starting with "6" and must end with a digit) was used to exclude any entries following the 5-digit postcode format. Anything not following this format could be printed off and examined to see how they can be updated: Zip code cleaning involved

1. Removing the 4 digit postcode suffix.
2. Removing 'IL' state letters from postcode
3. Removing zip codes those did not belong to Chicago area
4. Removing invalid 3 or 4 digit numbers holding the zipcode values

After a zip code cleaning regex function was applied, our database got rid of all invalid and inconsistent format of zipcodes.

Data Overview

After the data(OSM XML format) was passed through auditing and cleaning process using different functions (4audit.py, 5data.py, 6postcodeupdate.py) it was converted and stored in csv file formats. In the Next step, data in csv format was imported as tables to SQLite database, mydb.db, following the given schema. Data was explored with the SQL queries and additional information was collected.

```
File Sizes:
mydb.db..............................: 853M
nodes.csv............................: 540M
nodes_tags.csv.......................: 7M
sample.osm...........................: 150M
ways.csv.............................: 63M
ways_nodes.csv.......................: 171M
ways_tags.csv........................: 176M
```

Number of nodes:

```
cur.execute('SELECT COUNT (*) FROM nodes')

6304534
```

Number of ways:

```
cur.execute('SELECT COUNT (*) FROM nodes')

996230
```

Total number of users

```
cur.execute('''SELECT COUNT (*) as num
FROM (SELECT user FROM nodes UNION ALL
SELECT user FROM ways)''')

7300764
```

Top 10 contributing users

```
cur.execute('''SELECT user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways)
GROUP BY user
ORDER BY num DESC
LIMIT 10;''')


chicago-buildings  5603823
Umbugbene          604045
Oak_Park_IL        101175
Chicago Park District GIS  68077
Zol87              62141
bbmiller           61421
NE2                57155
asdf1234           39083
nickvet419         33651
alexrudd (NHD)     32894
```

Number of distinct/unique contributors

```
cur.execute('''SELECT COUNT (DISTINCT uid)
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways);''')

1666
```

Number of one time contributors

```
cur.execute('''SELECT COUNT(*)
FROM
(SELECT user, COUNT (*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways)
GROUP BY user
Having num = 1) ;''')



432
```

Additional Data Exploration

Top 10 Common Ammenities:

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




Place_of_worship 1682,
Restaurant 1077
School 890
Fast_food 473
Bicycle_rental 365
Cafe 338
Bench 290
Bar 220
Bank 184
Drinking_water 177
```

Popular Cuisines

```
cur.execute('''SELECT value, COUNT (*) as num
FROM nodes_tags
WHERE key = 'cuisine'
GROUP BY value
ORDER BY num DESC
LIMIT 10''')

Mexican  123
Sandwich  115
Pizza  104
Burger  92
Coffee_shop  89
American  70
Chinese  45
Italian  45
Thai  31
Japanese 19
```

Place of worship

```
cur.execute('''SELECT value, COUNT (*) as num
FROM nodes_tags
WHERE key = 'religion'
GROUP BY value
ORDER BY num DESC
LIMIT 10;''')


christian  1605
Jewish   53
Buddhist  6
Muslim  4
Unitarian_universalist 1
```

Conclusion and Recommendations

Exploration of data with SQL query revealed places of worship (e.g; church, temple etc.) accounted for the majority of the ammenities in Chicago area. Moreover, their number outnumbered the number of Schools in the area and this was indeed surprising observation.

On comparing the restaurent, Mexican cuisine was seen to be in the top of the list.

Thus, data auditing, cleaning and correction was done in OpenStreetMap data of Chicago,IL. Couple of categories were audited and cleaned as per the requirement of this excersise, however there is still major area that needs improvement. Invalid data, typos, incorrect and inconsistent values for the keywords were the few types of errors that was observed most of the times.

Since OpenStreetMap project is a global project which is coordinated by the volunteer users all over the world, if data is parsed through a standard programming system for every word for validity and consistency before importing it as a input, data quality of OpenStreetMap project could be improved.

Additional Suggestion and Ideas

--To minimize human errors there is a need of a data monitoring system. System should establish guidelines so only valid data and consistent script and terminologies are allowed as input.

--In todays world OpenStreetMap offers the free, openly available infrastructure data which has been utilized during responding to natural disasters, public health emergencies, and other crises in the world.

--Major Challenge is to spread information and attract more people to contribute in this project.

REFERENCES:

https://en.wikipedia.org/wiki/ZIP_Code (https://en.wikipedia.org/wiki/ZIP_Code)

https://en.wikipedia.org/wiki/XML (https://en.wikipedia.org/wiki/XML)

https://discussions.udacity.com/t/updating-postal-code/245757/18 (https://discussions.udacity.com/t/updating-postal-code/245757/18)

https://www.sqlite.org/faq.html (https://www.sqlite.org/faq.html)

https://discussions.udacity.com/t/unable-to-open-tables-and-run-queries-in-sqlite/307533/4 (https://discussions.udacity.com/t/unable-to-open-tables-and-run-queries-in-sqlite/307533/4)

https://mapzen.com/data/metro-extracts/your-extracts/f5b03d19a3d2 (https://mapzen.com/data/metro-extracts/your-extracts/f5b03d19a3d2)

https://discussions.udacity.com/t/csv-to-sqlite-import-issue/174730/39 (https://discussions.udacity.com/t/csv-to-sqlite-import-issue/174730/39)

http://www.webpages.uidaho.edu/~stevel/504/LearnPythonTheHardWay.pdf (http://www.webpages.uidaho.edu/~stevel/504/LearnPythonTheHardWay.pdf)