Data Wrangling Using OpenStreetMap

Max Sydow, WGU BS Data Management and Data Analytics, Udacity Data Analyst Nanodegree

Introduction

The central portion of the city of Minneapolis, Mn was extracted from the OpenStreetMap Overpass API as an XML file. I chose this city, because I grew up there and am missing it's mild summers. Data such as street names, and zip codes can be extracted from tags in the XML file using Python's iterparse() method. The data as elements inside the tags were used to populate csv's. These csv's were then used to populate tables in a SQL database using Python's sqlite3 package. From here the data can be explored using queries.

The link for the map is: https://www.openstreetmap.org/relation/136712

Preparing the Data

I used Jupyter notebooks for my Python scripts. The data.ipynb file contains a shape_element function that parses through tags and creates dictionaries which are passed to the process_map function that creates the csv's. The functions in audit.ipynb parse elements to pick out street names and zip (post) codes. The update.ipynb file contains functions to update street names and zip codes. Regular expressions and a mapping dictionary were used in the update functions to correct inconsistencies. Some such inconsistencies noticed include:

The use of abbreviations for street names, such as: Av, Av., and Ave for Avenue. I decided it would be more consistent to use the full word 'Avenue'.

The standard zip code format in the US is 5 digits, but I found one incomplete 3-digit zip code, and removed it. I also decided to remove the -xxxx, 4 digit extensions.

These cleaning updates were performed during dictionary creation in the shape_element function. Nodes and ways tags include child tags for street names and zip codes, these were the tags that the cleaning functions operated on.

A sample of the osm file was created using a script in the data.ipynb file, and is included.

Data Summary

The size of the Minneapolis.osm file is 66.9MB, and sample.osm is 8.56 MB. The sizes of the individual csv's are as follows:

Nodes.csv	23.7 MB
Nodes_tags.csv	2.98 MB
Ways.csv	2.94 MB
Ways_nodes.csv	9.95 MB
Ways_tags.csv	4.85 MB

The summary of tag counts using Python:

```
{'bounds': 1,
 'member': 49664,
 'meta': 1,
 'nd': 335597,
 'node': 269731,
 'note': 1,
 'osm': 1,
 'relation': 739,
 'tag': 200062,
 'way': 45657}
```

The data base consists of the tables: Nodes, Ways, Nodes_Tags, Ways_Tags, and Ways_Nodes. The entity relationships can be described as follows:

Nodes has id as primary key

Ways has id as primary key

Nodes_Tags has id as foreign key referencing the Nodes table primary key

Ways_Tags has id as foreign key referencing the Ways table primary key

Ways_Nodes has id as foreign key referencing the Ways table primary key

and node_id as foreign key referencing the Nodes table primary key

The entries original csv's appeared with a b' at the beginning and a ' at the end of each element. This format did not work well when creating the tables so a cleaning script was made before insertion. For example, the first entry in the first row of nodes1.csv is: b'33295121'. The new file, nodes.csv, contains cleaned entries, e.g. 33295121.

Querying the Nodes and Ways tables resulted in the same number of 'node' and 'way' tags as from the Python summary. Queries and results can be found in the sql.ipynb file.

There are 627 unique users that contributed to this XML file, 128 users made only one contribution. The top 10 contributing users are summarized as:

Username	Number of Appearances
Mulad	78844
iandees	66917
sota767	15770
Arun Balaji	13191
DavidF	12344
houston_mapper	11305
neuhausr	11008
nickrosencrans	8711
GOKUL606	8681
samhandler	6997

Further Exploration

I found it interesting to find the 5 most frequently appearing amenities.

Amenity	Count
Restaurant	291
Bicycle_parking	203
Bench	136
Café	100
Bicycle_rental	97

One neat feature on the map page is a tool called “query features” in the toolbar on the right-hand side. It allows one to select specific locations on the map and will list a set of amenities in the vicinity. Specific amenities, such as the name of a restaurant can be selected and its exact location is shown. More details such as node id, and user id are shown. Some of these entries are over 8 years old. Perhaps an algorithm could be designed to look for timestamps, then cross-reference with more updated sources to correct outdated information.