OpenStreetMap Data Case Study

Map Area

Lebanon, TN, United States

https://www.openstreetmap.org/search?query=lebanon%2C%20tn#map=12/36.1869/-86.3491

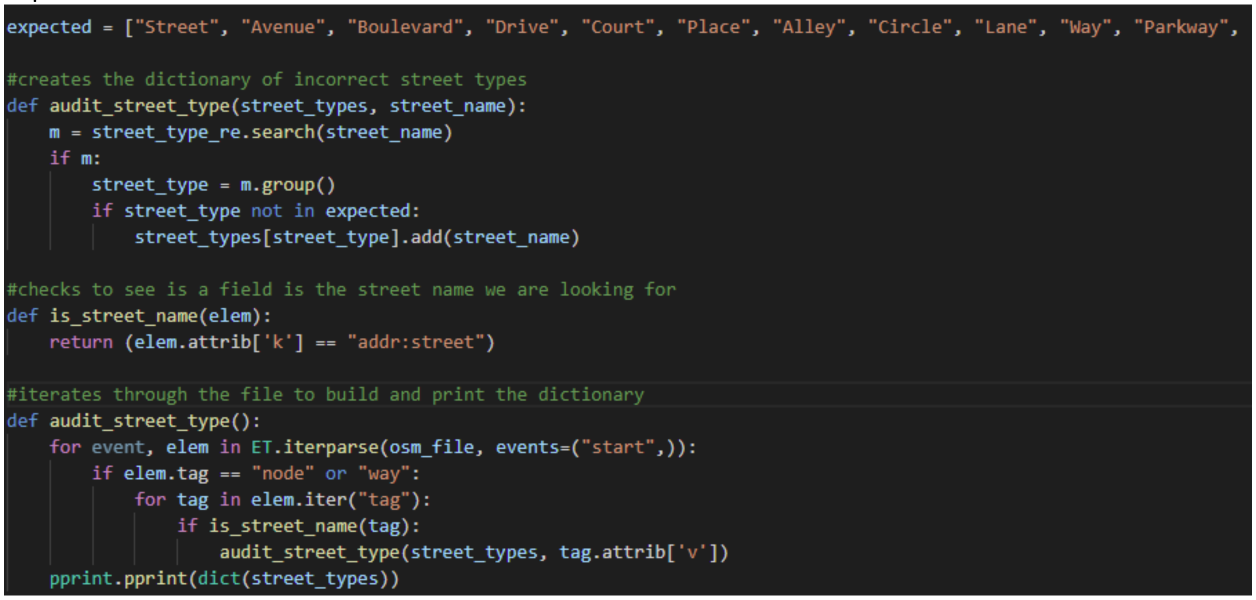
This map area is for an area that I live and work near, I wanted to see what kind of work had been performed on an area that is not a large metropolitan area.

Problems Encountered

As I started to audit my data selection following the case study, I first noticed that most of the street types were accurate, however there were a few streets that could be cleaned up to match the others. One example of this would be in a street name that read as "E Main St,". The second errors that I found initially fell into the county names for the data. Almost all the county names included the county and the state as in "Wilson, TN", which is not the name of the county only.

Fixing the Street Types

To fix the street names, I first created a dictionary of all the different street types, and then created an expected name list from this audit to include the corrected names.



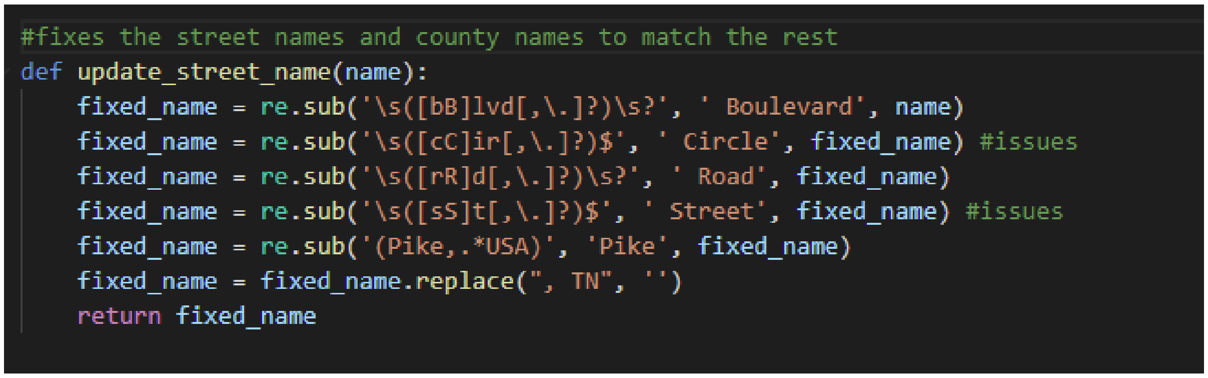
```
expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Alley", "Circle", "Lane", "Way", "Parkway",

#creates the dictionary of incorrect street types
def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)

#checks to see is a field is the street name we are looking for
def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")

#iterates through the file to build and print the dictionary
def audit_street_type():
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == "node" or "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    pprint.pprint(dict(street_types))
```

I then created some regular expressions to handle the corrections, since there were only 5 street types to fix in total. To save my program some time, I thought that it would be quicker to just fix the 5 types that were incorrect instead of iterating over every single street type and fixing something that wasn't broken.



```python
#fixes the street names and county names to match the rest
def update_street_name(name):
    fixed_name = re.sub('\s([bB]lvd[,\.]?)\s?', ' Boulevard', name)
    fixed_name = re.sub('\s([cC]ir[,\.]?)$', ' Circle', fixed_name) #issues
    fixed_name = re.sub('\s([rR]d[,\.]?)\s?', ' Road', fixed_name)
    fixed_name = re.sub('\s([sS]t[,\.]?)$', ' Street', fixed_name) #issues
    fixed_name = re.sub('(Pike,.*USA)', 'Pike', fixed_name)
    fixed_name = fixed_name.replace(", TN", '')
    return fixed_name
```

Once this finishes, all the street names for both the node and way tags match.

County Names

At first, I tried to find a data set where I could try to clean the zip code of each tag, however, finding a data set that I related to and had dirty zip codes proved to be very difficult. Instead, I chose to replace all the county names by stripping away the ", TN" values in the names and leaving just the county names. For example, "Wilson, TN" would become "Wilson". This was easy to accomplish with the above code. I chose to just include the solution to this problem in the street type function since this would run during the normal process of the code.

Database Queries

Size of the Files

Lebanon Map	54 MB
nodes.csv	20.7 MB
nodes_tags.csv	.35 MB
ways.csv	1.8 MB
ways_nodes.csv	6.8 MB
ways_tags.csv	3.3 MB
Data Wrangling.db	240 MB

Number of Unique Users

Using the following query:

```sql
SELECT DISTINCTROW Nodes.uid, Ways.uid, Count(*) AS [Count Of Nodes]
FROM Nodes INNER JOIN Ways ON Nodes.[uid] = Ways.[uid]
GROUP BY Nodes.uid, Ways.uid;
```

The number of unique users in the whole dataset is 719.

Number of Nodes and Ways

Using the following query:

```
SELECT Count(*) from Nodes;
```

The number of nodes in this database is 251,386.

Using the following query:

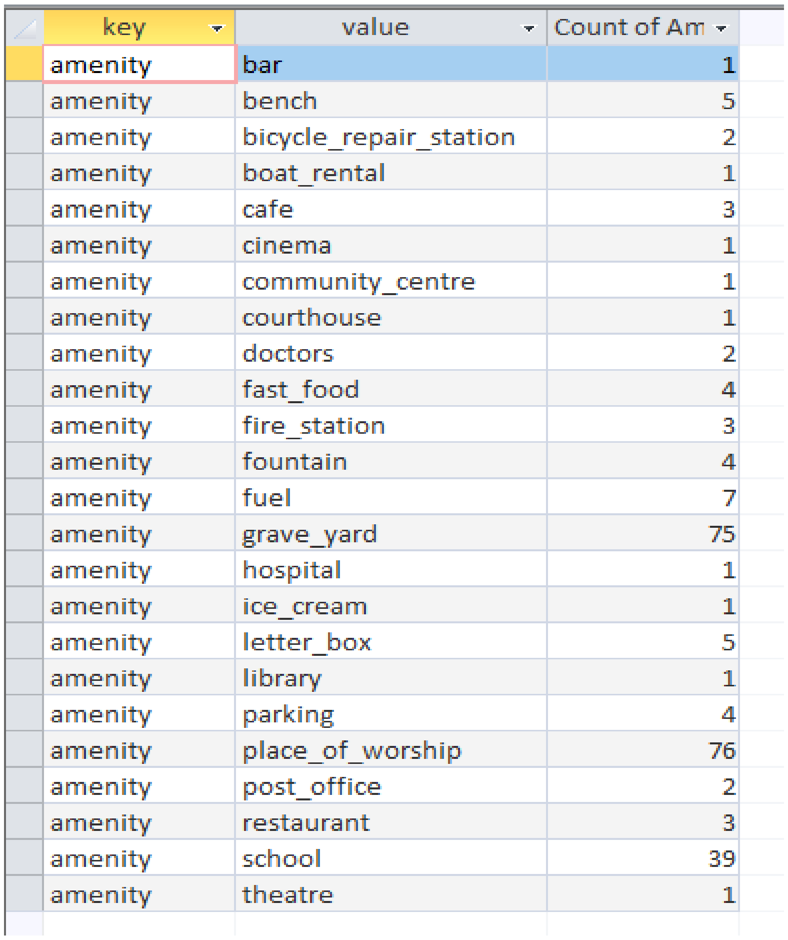
```
SELECT Count(*) from Ways;
```

The number of ways in this database is 29,721.

Number of Chosen type of nodes

Using the following query, I investigated the different types of amenities that this dataset offered and found the following results:

```
SELECT key, value, Count(*) as [Count of Amenity]
FROM Nodes_tags
WHERE key = 'amenity'
Group By key, value;
```



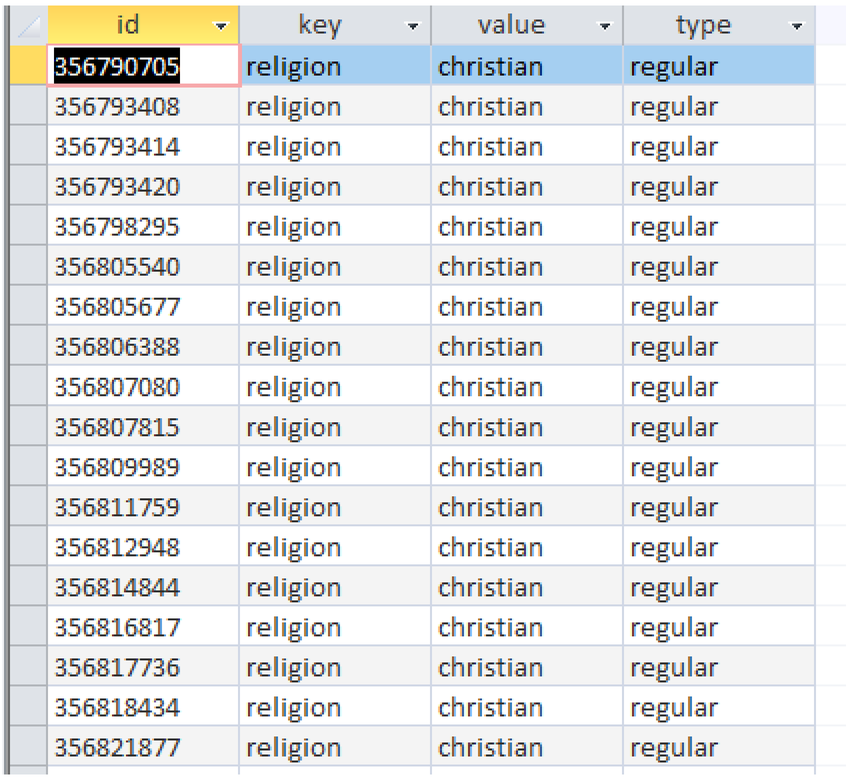
key	value	Count of Am
amenity	bar	1
amenity	bench	5
amenity	bicycle_repair_station	2
amenity	boat_rental	1
amenity	cafe	3
amenity	cinema	1
amenity	community_centre	1
amenity	courthouse	1
amenity	doctors	2
amenity	fast_food	4
amenity	fire_station	3
amenity	fountain	4
amenity	fuel	7
amenity	grave_yard	75
amenity	hospital	1
amenity	ice_cream	1
amenity	letter_box	5
amenity	library	1
amenity	parking	4
amenity	place_of_worship	76
amenity	post_office	2
amenity	restaurant	3
amenity	school	39
amenity	theatre	1

Based on this result, it looks like there is a staggeringly high number of churches or church related buildings, and graveyards. My next question for this query is to dive into the data further and determine if there is overlap in these two amenities and check if there are some locations that are counting in both rows.

Additional Ideas

Using the following query:

Select * from Nodes_tags where key = 'religion';



id	key	value	type
356790705	religion	christian	regular
356793408	religion	christian	regular
356793414	religion	christian	regular
356793420	religion	christian	regular
356798295	religion	christian	regular
356805540	religion	christian	regular
356805677	religion	christian	regular
356806388	religion	christian	regular
356807080	religion	christian	regular
356807815	religion	christian	regular
356809989	religion	christian	regular
356811759	religion	christian	regular
356812948	religion	christian	regular
356814844	religion	christian	regular
356816817	religion	christian	regular
356817736	religion	christian	regular
356818434	religion	christian	regular
356821877	religion	christian	regular

This query returns a list of every value in the node tags to look for each religion type. In this query, 76 results are returned, and each one of these results are Christian. However, as a former teacher in this area, I know many students practice other religions other than Christian, such as Muslim. Therefore, this dataset is not complete, and could use some improvements.

Conclusion

In conclusion, this area that I chose was relatively well cleaned before I investigated the data. There were only a handful of abnormal street abbreviations across 29,000 ways in this area. Also, I could not clean the area code information as many others do with this type of project since there were no abnormal formats. This leads me to wonder whether there have been some automated or scripted updates performed on the website in general that standardizes the area codes so users can't deviate from a set formula. Therefore, I had to clean the county names in the data to remove the state information from the county. I did this solely to show that I could find and replace select pieces of information, and I realize that cleaning has very little value on the overall dataset. In addition to those thoughts, investigating this dataset shows how rural the area is based on the types of amenities. This is an area where only one or two of any type of business can truly thrive, and that is reflected in the amenity query, where there are low numbers of amenities except the schools and religious institutions.

However, based on my investigation one improvement that could be made to the dataset would be to update some of the information in the data. Living in this area, I know that there are several fast-food restaurants, and looking at the data retrieved in this data set there are only 10 unique buildings that could act as a restaurant. There are more than 10 different fast-food restaurants in this area, so clearly this data is not accurate. One way that this data could be added to would be to encourage new data additions and active data accuracy checks through some form of gamification. If a system could be added to encourage cleaning in the form of a leaderboard, badges, achievements, or ranking system this could appeal to a younger or more competitive audience and encourage more accurate data. This would require significant up-front costs for front end designs such as the badge design or leaderboard/ranking layout and would require some form of back-end process to keep an accurate account of these rankings. However, if the gamification idea could be molded correctly, I know that teachers in the area would use this system to teach a curriculum skill. This could be in the form of keyboarding practice with a younger audience, or coding skills with a high school audience, and the fact that this system would still be open source, there would not be a lack of teachers who would want to integrate a real-world system into their classroom. Another way we could improve the accuracy of this data would be to cross reference the data being entered into the system with some form of API such as Google Maps API. Google Maps has one of the most accurate databases for location data that is publicly used. If we could integrate their API to use on the back end to prevent incorrect data from being entered, we could ensure better data quality and prevent a user from entering mass amounts of incorrect data just to move up the leaderboard.

Google Maps API Information:

https://cloud.google.com/maps-platform