Open Street Data Project

For this data set I wanted to take a look at the metropolitan area of Raleigh, NC. The OpenStreetMap partner Metro Extracts didn't have a defined area for Raleigh yet and the more popular choice on their site included the local towns of Durham, NC and Chapel Hill, NC. These two towns, along with Raleigh, comprise what is known as the Triangle and if one wanted to do a study of the whole area they should definitely be included.

But I wanted to solely look at Raleigh for several reasons, the main one being that it was the largest of the three cities and would have the most data about it. The second is that I was more familiar with Raleigh as a whole than I was with Durham or Chapel Hill and hoped this would help me spot any glaring errors in the Data itself.

So in order to limit my data to Raleigh I bounded by Open Map data by (35.6411, 36.0402) for latitude and (-78.9958, -78.2941) for longitude. But a quick audit of the data would needed to be undertaken to see if the boundaries helped keep me primarily to Raleigh or if the surrounding areas would slip in to the data

The first step to doing this was to look at the street data itself so I compiled a Python script called Audit.py that would parse through each 'Way' and 'Node' tag in the XML data and check their 'Tag' children and the values for the Address tags of the City, Street, State, Postcode. As I parsed through each the tags of the Way and Nodes the program checked to see if they referred to each of the address components if and if so then I would extract the value of each street, zip code, city, etc. and store it in either a set or defaultdict comprised of sets to prevent data duplication. The results showed that even in my attempt to limit it just to the city of Raleigh data some of the outside areas were able to slip in.

A quick look at some of the program output shows that the Street names have many of the same problems seen in the practice data mainly street suffixes applied erratically throughout the data. Different types of abbreviations with a variety of capitalizations that I will correct using code to break down the street string into a list using the .split() method and comparing the elements and see if they are in the new mapping dictionary for each street type. If so I will convert that part of the string to the full street type name before rejoining the string and inserting it into the new dictionary to write to my SQL server and CSV files.

Instead of just editing the last value I went through the street data like this in order keep keep addresses with suite numbers intact and not change those as they are often correctly appended to the end of the street name.

Looking at the Zipcode data shows me that the surrounding areas outside of the Raleigh city limits are included as several 275 and 277 starting zip codes were included when all zipcodes inside the Raleigh city limits begin with. Here is a brief selections of the types of zip codes that turned up that were not in the expected list:

{'2612-6401',
 '27069-5439',
 '27162-2750',
 '27162-3449',
 '27162-3452',
 '27162-3454',
 '27502',
 '27502-3918',
 '27502-3919',
 '27502-3922',
 '27511',
 '27511-4701'

Many of the zipcodes failed the audit because they have the four digit extensions on them. To clean the data I will strip off these extensions and any other extraneous characters such as letters to make the zip codes more uniform. I will keep all of the zip codes in the data set and not exclude them because the majority of them fall within the Raleigh city limits as is shown in my audit of the city data. Except for the first one in the example '2612-6401' which

doesn't have sufficient digits to be a zip code and could be several different possibilities. I decided to exclude this value as I had now way of accurately determining what zip code it could possibly be.

The city data has Raleigh as the number one location of the nodes by a wide margin(6,521 to 2,951 with the second largest the city of Cary is so close to Raleigh its almost part of the city) so my attempt to mainly focus on the Raleigh area is still successful. However, as with street names there are different versions of the cities and even some misspellings that I will need to correct as well.

Once the data was inserted in the SQL database I did a quick look at the zip code data using these two commands '**select value, count(*) from nodes_tags where key = 'postcode' group by value order by count(*) desc;'** and '**select value, count(*) from ways_tags where key = 'postcode' group by value order by count(*) desc;**' which returned small enough sets where I could quickly parse that the cleaning had worked correctly and that no tag with the key 'zipcode' had a 'value' that was longer than five digits as shown in the excerpts below:

nodes_tags:

27701|165
27513|149
27502|111
27511|97
27560|65
27587|60
27519|51
27601|47
27540|44
27705|43
27707|39

ways_tags:

27609|5378
27612|2354
27604|2350
27560|1820
27615|1279
27519|992
27705|980
27701|944
27713|856
27610|829
27511|827
27613|799
27703|793

The next issue was the city value of the tags. I set "Raleigh" as the Value I wanted to test against in my Audit.py file and it returned every 'city' tag that wasn't that value which resulted in this:

```
{' Raleigh': 1,
  'Apex': 516,
  'Apex, NC': 2,
  'Bonsal': 2,
  'Cary': 2910,
```

'Chapel Hill': 2,
'Clayton': 14,
'Durham': 1332,
'Garner': 7,
'Holly Springs': 47,
'Knightdale': 7,
'Louisburg': 1,
'Morrisville': 1622,
'Ralegh': 1,
'Ralegih': 1,
'Raleigh': 6390,
'Research Triangle Park': 5,
'Rolesville': 1,
'Wake Forest': 13,
'Wake forest': 1,
'Wendell': 3,
'Youngsville': 5,
'Zebulon': 3,
'cary': 1,
'durham': 2,
'raleigh': 3,
'wake Forest': 1})

As you can see there are several cities that have capitalization issues or spelling issues within the data and like with the street and state data I mapped the incorrect versions to the correct spellings in a dictionary and changed them as the program parsed through the data before inserting it into the SQL data base.

After insertions I performed this query '**SELECT tags.value, COUNT(*) as count FROM (SELECT * FROM nodes_tags UNION all SELECT * FROM ways_tags) tags WHERE tags.key = 'city' GROUP BY tags.value ORDER BY count DESC;'** which gave me these results:

Raleigh|6521
Cary|2951
Morrisville|1653
Durham|1370
Apex|528
Holly Springs|47
Wake Forest|16
Clayton|14
Garner|7
Knightdale|7
Research Triangle Park|5
Youngsville|5
Zebulon|4
Wendell|3
Bonsal|2
Chapel Hill|2
Louisburg|1
Rolesville|1

And it is clear here that the cleaning worked to great effect to standardized the spelling and capitalization of the cities in the data set.

While looking at the state data I noticed some tags where listed as being in Virginia instead of North Carolina. This was the result of my Audit.py file when I looked at the values of the 'state' key:

{'NC-', 'VA', 'nc', 'US', 'NC'}

There must be some sort of human error behind that because while North Carolina and Virginia border each other the border is a good hour and a half away from Raleigh. Some others were listed as US as well. Unable to tell from my initial audit what exactly was the problem I left these tags in to search for them later in the sql server to see what exactly happened.

When the SQL database was setup I was able to query to see what was going on with these values and used this command **SELECT * FROM nodes_tags JOIN (SELECT * FROM nodes_tags) nodes ON nodes_tags.id= nodes.id WHERE nodes_tags.value='VA';**' which resulted in this output:

367992481|state|VA|addr|367992481|building|yes|regular
367992481|state|VA|addr|367992481|county_name|Charlottesville (city)|gnis
367992481|state|VA|addr|367992481|ele|129|regular
367992481|state|VA|addr|367992481|feature_id|2073411|gnis
367992481|state|VA|addr|367992481|import_uuid|57871b70-0100-4405-bb30-88b2e001a944|gnis
367992481|state|VA|addr|367992481|name|Mabel Foster Clark Hall|regular
367992481|state|VA|addr|367992481|reviewed|no|gnis
367992481|state|VA|addr|367992481|source|USGS Geonames|regular
367992481|state|VA|addr|367992481|state|VA|addr
367992491|state|VA|addr|367992491|building|yes|regular
367992491|state|VA|addr|367992491|county_name|Charlottesville (city)|gnis
367992491|state|VA|addr|367992491|ele|134|regular
367992491|state|VA|addr|367992491|feature_id|2073405|gnis
367992491|state|VA|addr|367992491|import_uuid|57871b70-0100-4405-bb30-88b2e001a944|gnis
367992491|state|VA|addr|367992491|name|William Kerchof Hall|regular
367992491|state|VA|addr|367992491|reviewed|no|gnis
367992491|state|VA|addr|367992491|source|USGS Geonames|regular
367992491|state|VA|addr|367992491|state|VA|addr

So two random buildings from Charlottesville, VA are located in the Raleigh data somehow. Judging by the names of the buildings it appears that they are buildings located on the campus of the University of Virginia. How they ended up in this data set is unknown but there must be some error with the USGS Geonames system that labeled these as data belonging to this data set.

Besides those two values I mapped the other forms of North Carolina to a dictionary that would replace the abbreviations or other unusual spellings with "North Carolina". To check and see if the cleaning work I do a cursory search of the way_tags and nodes_tags tables where the key = 'state' with this search, '**SELECT tags.value, COUNT(*) as count FROM(SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) tags WHERE tags.key = 'state' GROUP BY tags.value ORDER BY count DESC;'** and get this output:

North Carolina|5366
NC|13
VA|2

NY|1
US|1

The US and VA were expected as I left them in the data set on purpose but the NY was completely unexpected as the original Audit did not catch it as it definitely wasn't in the expected values for states Dictionary. Also the fact that some 'NC' slipped past the cleaning will need to be looked at in the second pass through the data. Still 99.7% of the data is in the preferred form for the 'state' key and for the first pass through the data I'm happy with those percentages.

The last major problem with the address data is the use of the TIGER gps imports on some of the ways nodes that doesn't follow the same structure as the rest of the address data most notably . I'm going to attempt to piece together that data and place it in the same schema as the other tags before I insert it into the SQL database. If the tag had multiple names for the address I just used the first one and ignored any additional names.

In order to asign the TIGER Data to the more traditional I created a function that would stored the ways nodes into a list and then passed that list to a function I wrote that combined the split up street names into one value and pulled out the post code that I then assigned to the 'addr' type and gave them the same key names of 'street' and 'postcode' that the other more normally inserted had so that when I was making SQL queries on the database this data would show up in my searches.

With the data cleaned and inserted into my SQL database I was ready to run some SQL queries on the data. First an overview of each file size that resulted from the CSV/SQL creations:

- RaleighStreetData.osm 582MB
- nodes.csv 223MB
- nodes_tags.csv 2.4MB
- ways.csv 16MB
- ways_nodes.csv 71MB
- ways_tags.csv 36MB
- Street_Data.db 303MB

Next let's look at the size of the ways , nodes, and users:

- Number of Nodes: 2,874,786 using **SELECT count(*) FROM nodes;**
- Number of Ways: 292,584 using **SELECT count(*) FROM ways**;
- Number of Unique Users 988 using **SELECT count(distinct(users.uid)) FROM (select uid from nodes union all select uid from ways) users;**

Another thing I was interested in was the break down of natural features or landmarks used as nodes in the street data. So I used '**SELECT value, count(*) FROM nodes_tags where key='natural' GROUP BY value ORDER BY count(*) DESC;**' to look and see what I came up with:

- Tree: 1,141
- Peak: 4
- Water: 1

Raleigh isn't exactly known as a mountainous area so I was interested to what was considered a peak in this area. So I used the SQL command "**SELECT nodes_tags.value FROM nodes_tags JOIN (SELECT value, id from nodes_tags where value = 'peak') peak ON nodes_tags.id = peak.id WHERE nodes_tags.key = 'name';**" and came up with these two values:

- Adam Mountain
- Huckleberry Hill

I also wanted to look at which man made ways had the most nodes associated with them as well in order to see which roads have the most information about them in the Open Street Data. To do this I used this SQL command, '**SELECT name.value, count(ways_nodes.id) as count, type.value FROM ways_nodes JOIN (SELECT value, type, id FROM ways_tags WHERE key='name') AS name ON ways_nodes.id = name.id JOIN (SELECT value, id from ways_tags where key='highway') AS type ON ways_nodes.id = type.id GROUP BY name.value ORDER BY count DESC LIMIT 10;'** which gave me these results:

•Falls Lake Trail|4646|path
•Neuse River Trail|1490|cycleway
•Capital Boulevard|1462|trunk
•286 Trails|1283|path
•Hedingham Golf Course Path|1081|path
•New Bern Avenue|1001|trunk
•Walnut Creek Trail|819|cycleway
•Crabtree Creek Trail|757|cycleway
•Regency Trails|752|path
•Spring Forest Road|724|primary

I found it most unusual at first that the most data was gathered on trails and bike paths but upon further review that may be the areas where most people are able to gather the data on their gps because they have the extra time to think about it unlike when they would be driving.

Perhaps one of the best ways to improve this data set is to work on consolidating and cleaning up the TIGER data that is contained in the ways_tags tables. Even though I started that work when I parsed the data originally I still left in the original TIGER data and was only able to pull the street name and zip code. There are still the address number, county, state, values that can be pulled from the TIGER data and converted into an OSM 'addr' type that would make the database more uniform.

This would make the results of the database queries more complete and make the searches themselves cleaner as well to be implemented. One problem with doing this is that it might not be entirely possible to transition the TIGER data to the standard address structure completely with a program and may require some level of human interaction to make sure the data is transitioned properly. Especially given the fact that the ways in the TIGER data can have multiple names for each road.

Another issue would be to continue removing the extraneous zipcdoes that don't belong in the Raleigh city limits in order to get a data set more representative of the actual city limits. The main problem with this is that zip codes often crosss city limits and even creating a list of acceptable zip codes would include locations outside the actual city of Raleigh. Still if I was to continue to cleaning the data to make it more accurate this would be my next step in my next cleaning effort of the data.