OpenStreetMap Data for Austin, TX

Map Area

Austin, TX USA

https://mapzen.com/data/metro-extracts/metro/austin_texas/

This area is more familiar to me at the moment so I chose it. It also meets the project requirements of having at least 50 MB file size uncompressed. A sample of this file was generated using the code provided in the instructions for the project:

https://www.dropbox.com/s/084lnztuwgxdgtm/sample.osm?dl=0

Problems Encountered in the Map

Exploration of the sample osm file as well as the whole osm file showed that:

1. Street names need to be unabbreviated.
2. Inconsistent abbreviation for street names poses challenge in cleaning data (ex: IH35, I H 35, I-35, I35)
3. Phone number format is not consistent (ex: (512) 782-5659, +1 512-472-1666, 51224990093, 512 466-3937, etc.)
4. More than one phone number are entered in the field (ex: "Main: (512) 899-4300 Catering: (512) 899-4343")
5. Postcodes didn't have a consistent format--some have county codes, some do not.
6. City name format is not consistent (ex: Pflugerville, TX; Pflugerville)

Cleaning of Street Names

Results of the audit of street names using the method described in the case study exercises for the course showed that aside from some street names being heavily abbreviated, some street names are abbreviated inconsistently. In this project, the "update_name" function introduced in the case study exercises was modified by adding subfunctions and other lines of code, to successfully update the street names. Examples of updates done are:

```
North IH 35 => North Interstate Highway 35
Calhoun Ln => Calhoun Lane
FM 685 => Farm-to-Market Road 685
W. University Avenue,Ste 320 => West University Avenue Suite 320
```

The function was able to distinguish between some abbreviations:

```
Avenue H => Avenue H (H stayed the same)
N I H 35 Bldg 7 => North Interstate Highway 35 Building 7 (H was converted to Highway)
```

However, after the updates, there were still some problems remaining after the clean up, such as certain streets have different names. For example, Ranch Road 620 is also referred to as Farm-to-Market Road 620, US Highway 290 is also Country Road 290. These were not addressed in the project although it could be easily added to the "mapping_street" dictionary used by the function.

The "clean" Function

Instead of using individual functions which were written to fix different problems in the street name, a "clean" function was created and used in the "shape_element" function (discussed below).

```
def clean(value, tag, mapping_street, expectedcities, mapping_city):
    if is_street_name(tag):
        value = update_name(value, mapping_street)
    elif is_phone(tag):
        value = update_phone(value)
    elif is_postcode(tag):
        value = update_postcode(value)
    elif is_city(tag):
        value = update_city(value, expectedcities, mapping_city)
    return value
```

Extraction of Data from OSM File to CSV Files

Data were extracted from the OSM file using the functions from the case study exercises from the course. The "meat" of the process happens in the "shape_element" function which not only parses the osm xml data but also cleans the data using the functions discussed above. The general scheme for processing osm files start from creating csv files as output files using the codecs module, then shaping the output, validating this output against a set schema and then writing the output onto the csv files.

A problem I encountered in processing the whole osm file with validation set to True even if I didn't obtain any errors processing the sample file with validation. This was because the data that sets the error on is not present in the sample file. Only when I analyzed the output files was I able to figure out what's wrong with my function. In short, the problem is caused by the absence of a line that allows the code to ignore problematic characters for values of the attribute 'k':

```
try:
    problem_chars.search(tag.attrib['k']).group()
except AttributeError:
   .....
```

This would have worked if I included "continue" in the third line of the code.

```
try:
    problem_chars.search(tag.attrib['k']).group()
    continue
except AttributeError:
    ....
```

I however changed my code to use the if/else statement but still it needed the "continue" statement for the code to work.

Creation SQL Database

Creating the SQL database (atx_osm.db) was done using Python according to the method outlined in the course forum (https://discussions.udacity.com/t/creating-db-file-from-csv-files-with-non-ascii-unicode-characters/174958/6), using the schema specified in the following site: https://gist.github.com/swwelch/f1144229848b407e0a5d13fcb7fbbd6f. The process was straightfoward. All codes are contained in this notebook: https://github.com/mudspringhiker/wrangle_open_streetmap_data/blob/master/db_creation.ipynb.

Querrying the SQL Database

Querrying for list of cities showed that pretty much of all the cities were cleaned:

```
cities = cur.execute("""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""").fetchall()
print cities

Out: [(u'Austin', 3068),
    (u'Round Rock', 113),
    (u'Kyle', 64),
    (u'Cedar Park', 43),
    (u'Pflugerville', 37), .....
```

However, looking at the postcodes (querried using a similar code as that used for querrying for cities), there were three "None" values. To figure out what these should be, I querried for the accompanying information with these values.

```
missing_postcodes = cur.execute("""SELECT *
                                   FROM (SELECT * FROM nodes_tags
                                     UNION ALL
                                     SELECT * FROM ways_tags) tags
                                   WHERE tags.key = 'postcode'
                                   AND tags.value = 'None'""").fetchall()
```

```
print missing_postcodes

Out:    [(2152207067, u'postcode', u'None', u'addr'),
        (247506590, u'postcode', u'None', u'addr'),
        (383791236, u'postcode', u'None', u'addr')]
```

To determine what info is accompanying id 2152207067, the following query was done:

```
cur.execute("""SELECT *
         FROM (SELECT * FROM nodes_tags
         UNION ALL
         SELECT * FROM ways_tags) tags
         WHERE tags.id = 2152207067""")
cur.fetchall()
Out: [(2152207067, u'name', u'Nyle Maxwell - Taylor', u'regular'),
    (2152207067, u'shop', u'car', u'regular'),
    (2152207067, u'website', u'www.nylemaxwellcjd.com', u'regular'),
    (2152207067, u'street', u'United States Highway 79', u'addr'),
    (2152207067, u'postcode', u'None', u'addr')]
```

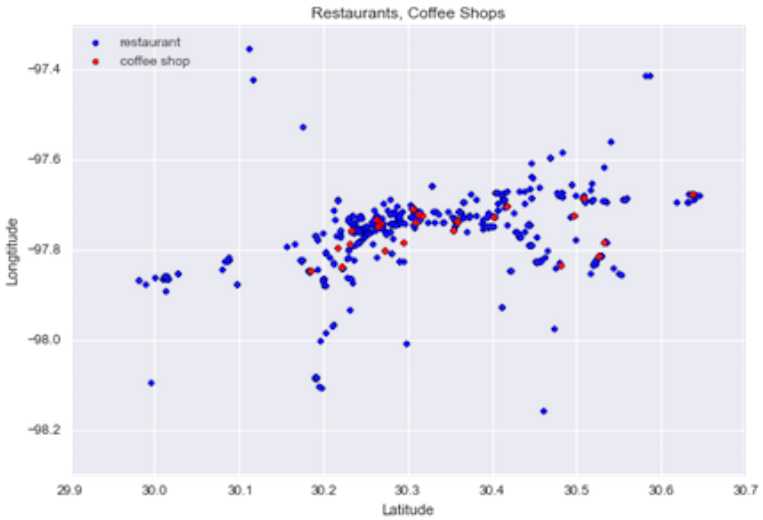
From this result and accessing the provided website, it can be found that the postcode should be 76574. The other missing postcodes were determined in the same way.

Locations of Restaurants

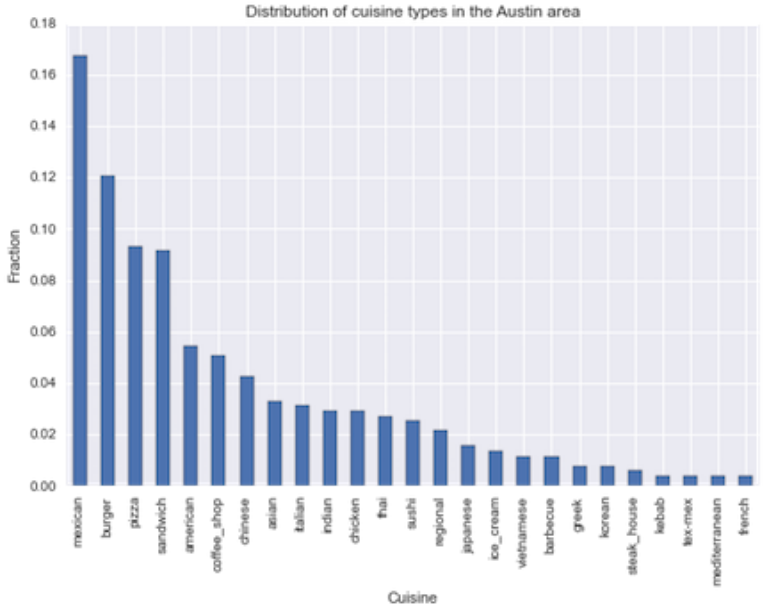
The query used to obtain a list of all the restaurants in the Austin, TX area was:

```
cuisine_loc = cur.execute("""SELECT b.id, b.value, nodes.lat, nodes.lon
                             FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags) b
                               JOIN nodes ON b.id = nodes.id
                             WHERE b.key = 'cuisine'""").fetchall()
```

Obtaining the locations of the coffee shops will then have a similar code and plotting the locations of these restaurants vs. the locations of coffee shops can then be done:



Lastly, querrying the database for the most popular cuisines was done. Pandas was used to eventually plot the distribution of the different types of restaurants in the Austin, TX area. It is no surprise that the area has a lot of Mexican restaurants.



Data Overview and Additional Ideas

The following are some information about the dataset:

File Sizes

```
austin_texas.osm        1.41 GB
atx_osm.db              820.4 MB
nodes.csv               604.3 MB
nodes_tags.csv          11.7 MB
ways.csv                48.6 MB
ways_tags.csv           70.6 MB
ways_nodes.csv          175.6 MB
```

Number of Nodes

```
In [17]: cur.execute("SELECT COUNT(*) FROM nodes")
         nodes = cur.fetchall()
         nodes
Out[17]:
         [(6356394,)]
```

Number of Ways

```
In [18]: cur.execute("SELECT COUNT(*) FROM ways")
         ways = cur.fetchall()
         ways
Out[18]:
        [(666390,)]
```

Number of Users/Contributors

```
In [19]: cur.execute("""SELECT COUNT(DISTINCT(e.uid))
                        FROM (SELECT uid from nodes UNION ALL SELECT uid FROM ways) e""")
         users = cur.fetchall()
         users
Out[19]:
         [(1146,)]
```

Top 10 Contributing Users

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

Output:

```
[(u'patisilva_atxbuildings', 2743705),
 (u'ccjjmartin_atxbuildings', 1300514),
 (u'ccjjmartin__atxbuildings', 940070),
 (u'wilsaj_atxbuildings', 359124),
 (u'jseppi_atxbuildings', 300983),
 (u'woodpeck_fixbot', 223425),
 (u'kkt_atxbuildings', 157847),
 (u'lyzidiamond_atxbuildings', 156383),
 (u'richlv', 50212),
 (u'johnclary_axtbuildings', 48232)]
```

However by using the pandas module, a better looking table of the results can be obtained:

Out[29]:

	users	count
0	patisilva_atxbuildings	2743705
1	ccjjmartin_atxbuildings	1300514
2	ccjjmartin__atxbuildings	940070
3	wilsaj_atxbuildings	359124
4	jseppi_atxbuildings	300983
5	woodpeck_fixbot	223425
6	kkt_atxbuildings	157847
7	lyzidiamond_atxbuildings	156383
8	richlv	50212
9	johnclary_axtbuildings	48232

Suggestions for Improvement of Data

One aspect that always crop up during clean up of my data was loss of data, such as in the case where one of two phone numbers provided gets discarded. This may be remedied by using a list as value for the field. However, the validation check will flag this and create an error. A non-SQL database might be more applicable in handling this case.

Another problem with the data itself is the presence of more than one field names for one type of data. When the values of attribute 'k' was explored, there were at least two "fix me"'s as values. There were also more than one for phone numbers and postal codes. A standardization of the k values should be instituted by OpenStreetMap. Anything that does not fit the list of these k values should create an error upon data entry for contributors. Also, the format for the values might also be standardized. A disadvantage of such rules however, might discourage contributors causing a slow development of OSM. However, if an automated cleaning program is instituted, it might be ok.

Conclusion

Information from an xml file can be extracted for data by Python through the xml.eTree.ElementTree module. This can be converted to a csv file which can be converted to an sql database (or to a pandas dataframe, which is not shown here, but was explored in another unit of the course). SQL databases can be converted to a pandas dataframe.

Cleaning of data takes a while. Knowledge of the nature of data also is very important so the best decisions on what to do with it can be done.

Other References

Automate the Boring Stuff with Python: Practical Programming for Total Beginners, A. Sweighart, No Starch Press San Francisco, CA, USA ©2015 ISBN:1593275994 9781593275990

http://stackoverflow.com/questions/19877344/near-syntax-error-when-trying-to-create-a-table-with-a-foreign-key-in-sqlit

Brandon Rhodes - Pandas From The Ground Up - PyCon 2015, https://www.youtube.com/watch?v=5JnMutdy6Fw

Udacity Data Wrangling Course (https://classroom.udacity.com/nanodegrees/nd002/parts/0021345404/modules/316820862075460/lessons/491558559/concepts/816599080

About the xml module method .iterparse:

http://effbot.org/zone/celementtree.htm

http://effbot.org/zone/element-iterparse.htm