Wrangling OpenStreetMap Data by Rohit Vinnakota

1.Parsing the file

I used the xml python library to parse and read the OSM file. The count_tags function that counts the top level tags returns

{'node': 1, 'nd': 1, 'member': 1, 'tag': 2, 'relation': 1, 'way': 1, 'osm': 1}

After quickly scanning the data, I decided to tackle two problems and try and fix them programmatically. Auditing incorrect street names and correcting zip-codes. In order to audit the incorrect street names, I need to first obtain a list that would hopefully help me map the incorrect names to the correct ones. For this purpose, I decided to do some site-scraping on a website that already contains those names.

2. Using site scraping to obtain a list of valid street names

The url http://gimme-shelter.com/term-2/steet-types-designations-abbreviations-50006/ contains all the different unique street names in Edmonton. I used the beautiful soup library to obtain the information and iterated through the tags and read the different street names from the table on the website into a python list. This list can then be iterated over and cleaned using python functions. The final list looks something like:

[u'Abbey', u'Acres', u'Autoroute', u'Avenue', u'Bay', u'Beach', u'Bend', u'Boulevard', u'By-pass', u'Byway', u'Campus', u'Cape', u'Carr\xe9', u'Carrefour', u'Castle', u'Centre', u'Cercle', u'Chase', u'Chemin', u'Circle', u'Circuit', u'Close', u'Common', u'Concession', u'Corners', u'C\xf4te', u'Cour', u'Cours', u'Court', u'Cove', u'Crest', u'Cross', u'Crossing', u'Cul-de-sac', u'Dale', u'Dell', u'Diversion', u'Downs', u'Drive', u'\xc9changeur', u'End', u'Esplanade', u'Estates', u'Expressway', u'Extension', u'Farm', u'Field', u'Forest', u'Freeway', u'Front', u'Gardens', u'Gate', u'Gateway', u'Glade', u'Glen', u'Green', u'Grounds', u'Grove', u'Hall', u'Harbour', u'Haven', u'Heath', u'Heights', u'Highlands', u'Highway', u'Hollow', u'\xcele', u'Impasse', u'Inlet', u'Island', u'Keep', u'Key', u'Knoll', u'Lake', u'Landing', u'Lane', u'Limits', u'Line', u'Link', u'Lookout', u'Loop', ……]

3. Basic querying using python libraries

Before diving into cleaning the data programmatically, I will first take note of a couple of insights, the list of users(619) and the user with the most submissions (‘edmontongeo’, 2407878). It can be inferred that edmontongeo is most likely a bot or a company which would be appropriate. I will later use these values to test against my database queries when I do convert the current XML data into an SQL database.

4. Updating incorrect street names and auditing zipcodes

To audit zipcodes programmatically, I make use of the defaultdict libraries and helper functions to identify any street types that may be incorrect. The resulting set shows that there are many streets with numerical names(such as highway and avenue numbers and) and other streets with inconsistent or abbreviated names. A small example looks like:

'100': set(['100 Avenue', '100 Street', 'Sturgeon 100 Crescent']),
'100A': set(['100A Avenue', '100A Street']),
'101': set(['101 Avenue', '101 Street', '101 street']),……..
'Zodiac': set(['Zodiac Drive', 'Zodiac Way']),
'a': set(['49th a Street',
'51st a Avenue',
'52nd a Avenue',
'52nd a Street']),
'ave': set(['131 ave NW', '48 ave']),
'avenue': set(['117 street 87 avenue', '90th avenue']),
'calahoo': set(['calahoo Road']),
'herert': set(['herert road']),
'road': set(['herert road']),

To correct this, I created a mapping distionary that updates each incorrect key with a cor responding value that is found in the street list and updated all the incorrect street nam es. I do not consider numerical names to be incorrect. The resulting set looks something like

106 Street => 106 Street
107 Street => 107 Street
107 street => 107 street
107 Avenue => 107 Avenue

107 St => 107 Street
.
.
.
St Albert Trail North-west => St Albert Trail NW
Alexander Circle North-west => Alexander Circle NW
Richards Crescent North-west => Richards Crescent NW
Green Wynd North-west => Green Wynd NW
81 Avenue North-west => 81 Avenue NW

I chose to update all territorial names to be consistent rather htan having. For example: rather than having “Northwest”, “north-west”, “NW”, etc, I chose to convert them all int o just “NW”.

For incorrect zipcodes, I audited through all tags that contain zipcodes and deleted the c orresponding tags whose length is more than 6 or if it does not start with a “T”

5. Converting our data to prepare for insertion into an SQL database

5 files were generated in a csv format. These files are the nodes, nodes_tags, ways, ways_nodes, and ways_tags that each contain their corresponding osm_xml tags. These csv files can then be imported into an sqlite3 database to perform querying on. The files are relatively large and were first cleaned up using the re library to eliminate problematic characters.

FILE SIZES

edmonton_canada.osm => 784 MB

nodes.csv => 312 MB nodes_tags.csv => 18.95 MB

ways.csv => 28.05 MB

ways_tags.csv => 102.4 MB

ways_nodes.csv => 25.6 MB

p3main.db(the sql database) => 581 MB

6. Further insights through querying

Once the data is in the sqlite database, querying for the number of users yields

"SELECT count(DISTINCT user) FROM (SELECT user FROM ways UNION ALL SELECT user FROM nodes)";

=614

"SELECT user,count(user) FROM nodes GROUP BY user UNION SELECT user,count(user) FROM ways GROUP BY user ORDER BY count(user) DESC;"

= ('edmontongeo', 2119823)

As we can see, these are pretty close to the original values from section 3. I attribute the slight difference to our elimination of problematic characters.

However, the query to see the distinct city tags around Edmonton yields a messy list with multiple repetitions. The data appears very inconsistent with multiple "Edmonton" city values. This is one of the major problems with user submitted data. In addition to this, we can conclude that the map area used is not only that for Edmonton but for surrounding towns and counties as well. The resulting list looks something like

```
…… ('Gwynne', 8), ('Rich Valley', 8), ('Mearns', 6), ('Rivi
ere Qui Barre', 6), ('Villeneuve', 6), ('Camrose', 5), ('Lakevie
w', 4), ('Pigeon Lake #138A', 4), ('Winfield', 4), ('', 3), ('Ca
mrose', 3), ('Fort Saskatchewan', 3), ('edmonton', 3), ('Edmonto
n ', 2), ('Yellowstone', 2), ('Acheson', 1), ('Acheson ', 1), ('
Beaumont', 1), ('Devon', 1), ('Edmonron', 1), ('Edmonton ', 1),
('Edmonton International Airport', 1), ('Edmonton, AB', 1), ('Le
duc', 1), ('Morinville', 1), ('Smokey Lake', 1), ('Smoky Lake',
1), ('St Albert', 1), ('Tofield', 1), ('edmonton', 1)]
```

A lot of random querying such as (SELECT * FROM nodes_tags WHERE key= 'amenity') produced null results in the values column. This suggests further cleaning is required to obtain reasonable insights. Almost all nodes_tags have a key but no value. This shows a lack of full utilization of the OSM feature set or the existence of problematic characters. A major reason may be that Edmonton is not an extremely dense or populus city like New York or San Francisco thus has a lower user base constantly updating data. Some suggestions for improvement include a unified data entry scheme on the OSM website, where unnecessary tags are not created if the user does not fill in the value. This can work in a format similar to google docs, where the user fill out their pertinent

contribution in a form (name of the place, type of place, address, zipcode, etc.) and the data is inserted through a backend process which can also act as a auto-corrector to prevent the user from entering nonsensical data.

As it stands, there remains no initiative to have a unified format for the data. This can be fixed by automating a lot of self-correction and user entry processes to make the data that does pass through cleaner than raw user submitted data.

Sources

http://gimme-shelter.com/term-2/steet-types-designations-abbreviations-50006/

https://www.crummy.com/software/BeautifulSoup/bs3/documentation.html#The%20basic%20find%20method:%20findAll(name,%20attrs,%20recursive,%20text,%20limit,%20**kwargs)

http://stackoverflow.com/questions/13482586/how-can-i-tell-python-to-take-the-last-2-words-from-an-email-that-is-in-a-list-l

http://stackoverflow.com/questions/26871866/print-highest-value-in-dict-with-key

https://gist.github.com/carlward/54ec1c91b62a5f911c42#file-sample_project-md

All relevent coursework in Udacity Data Analyst Nanodegree(especially the Case:Study OSM and XML module)