Data Wrangling

Data Set

I chose to download the OpenStreetMap data for Seattle, Washington. I chose Seattle because it is the city that I am planning to move to in the next few years.

The link to the dataset is here: https://www.openstreetmap.org/relation/237385

Problems encountered

I ran initial audits on a sample of the data using the provided python script to create a sample data set. I found and ultimately corrected five issues with the data.

- Inconsistent/abbreviated street endings (156th Place NE, S Brandon St)
- Abbreviated street names (156th Pl Northeast, NE 5th Street)
- Zip codes in various formats (98101, 98310-3604)
- Incorrect zip codes (98033 (Kirkland zip code), W Lake Sammamish Pkwy NE)
- Addresses not in Seattle (city tag showing Bellevue, Kirkland, etc)

Inconsistent street endings

I adapted the code from the case study to gather all of the street endings that were in the sample file. With these I expanded the list of expected values, and a list of values that needed to be adjusted. These are both in the file adj_street_endings.py. Below is an example of the code I used to populate this list. “iterate_elements” and “get_node” are functions that are used in all of the audit scripts, so I added these to a separate module, so I wasn’t rewriting the same code.

```
for x in iterate_elements(osm_file):
    c = get_node(x)
    if c is not None:
        for y in c:
            if y.get("tag") is not None:
                if y.get("tag").get("attributes").get("k") == "addr:street":
                    addr = y.get("tag").get("attributes").get("v")
                    m = street_type_re.search(addr)
                    if m:
                        street_type = m.group()
                        if street_type not in expected:
                            y = addr.split(" ")
                            y[-1] = re.sub('[^A-Za-z0-9]+', '', y[-1])
                            dl[y[-1]] += 1
```

Abbreviated street names

After I corrected the street endings I adjusted the script to split all of the words for the address and put them into a set so I could see what was left. There were instances of abbreviated words, such as directions (NE), or descriptors (ste., pl) that were in the street name but were not at the end. I adjusted the code to fix the street endings to also fix these.

Zip codes in various formats and zip codes not in Seattle

Correcting the zip codes was much simpler than the street names. For simplicity, I just truncated all of the zip codes to be just the first five characters. This removed any of the zip codes with the additional four digits, or those that had more than five characters but were not a 5-4 zip code.

While searching the zip codes I also noticed that there were zip codes that were not for Seattle. I noticed this by checking the addr:city tag along with the addr:postalcode tag – many of the city tags showed Bellevue, Kirkland, etc. I first attempted to fix this by using the API at zipcodeapi.com and the requests module to search the city linked to the zip code – example in the code below. However, the API had a limit of 50 searches for hour, and there were more than 50 zip codes to search.

```
import requests
import json

req_string = 'https://www.zipcodeapi.com/rest/{}/info.json/{}/degrees'

for value in dl:
    if post_code_re.search(value):
        r = req_string.format(api_key,value)
        response = requests.get(r)
        x = response.json()
```

I ended up finding a list of all zip codes in Seattle and downloading this as a csv and importing it into a list in Python. I checked all zip codes against this list and did not return any records where the zip code was not for Seattle.

Addresses not in Seattle

After checking the zip codes, I decided to double check the city tag and validate that all of the records were for Seattle. I found that there were still addresses that were listed for cities other than Seattle. I adjusted the postal code script slightly to only return the record if the city was Seattle.

Ideas for additional improvements

Like in the sample project, I noticed that there were records from Tiger GPS that had street addresses in different formats than the standard OSM format. One potential improvement would be to write a script that finds these and adjusts them into the proper format. However, without an explanation of the Tiger format, this will be a difficult task, because it not perfectly clear from the data how these should be structured. For example, there is a tag called name_base that seems to have street names, but it also has records with more than one street name (e.g. 138th; Duvall). Additionally, there is also a name_base_1, name_base_2, and name_base_3. If you could determine how to convert these, however, it would go a long way to making all of the data more consistent.

Statistics and overview

I used MongoDB instead of SQL, since I use SQL every day in my day job. My statistics and queries will come from MongoDB.

Initial OSM file size: 977 MB

Final JSON file size: 1.28 GB

MongoDB collection size: 1.29 GB

Total Documents: 4,600,000

Index Total Size: 42.83 MB

Count of node records: 4,184,877 - col.count_documents({"node":{"$exists": True}})

Count of way records: 415,123 - col.count_documents({"way":{"$exists": True}})

Records with contact:facebook tag: 40 - col.count_documents({"node.children.tag.attributes.k":"contact:facebook"})

Records with contact:twitter tag: 37 –

col.count_documents({"node.children.tag.attributes.k":"contact:twitter"})

Unique contributing users: 2205 – col.distinct("node.attributes.user")

Top ten contributing users:

```
p = [{"$group": {"_id": "$node.attributes.user", "count" : {"$sum":1}}}
     ,{"$sort":{"count":-1}}
     ,{"$limit" : 10}]

x = col.aggregate(p)
```

Output:

```
[{'_id': 'Glassman', 'count': 834494},
 {'_id': 'SeattleImport', 'count': 659825},
 {'_id': None, 'count': 415123},
 {'_id': 'Omnific_Import', 'count': 392775},
 {'_id': 'Glassman_Import', 'count': 195533},
 {'_id': 'Omnific', 'count': 137274},
 {'_id': 'sctrojan79', 'count': 108107},
 {'_id': 'STBrenden', 'count': 98621},
 {'_id': 'seattlefyi', 'count': 92670},
 {'_id': 'seattlefyi_import', 'count': 77568}]
```