OpenStreetMap Data Case Study

Map Area

Beaver County, PA, US and minor outlying and surrounding areas

http://beavercountypa.gov/Pages/Default.aspx (http://beavercountypa.gov/Pages/Default.aspx)

This region is where I live, where I currently work from home, and volunteer as a firefigher & EMT. I'm curious how much data has been entered for this somewhat rural community.

Problems Encountered in Map Data

Below I created a sample subset and examined some of the data.

Problems encountered included:

1. Inconsistently abbreviated street types, including Blvd, Tpke, St, and ("Rd, Rd., rd") for Road, etc.
2. Zip codes formatted with more than 5 integers, such as: 15010-4503
   - Note - Tiger GPS data caused an issue with counting zip codes, which I'll explain below.
3. Problem characters within the listings, and tiger GPS data formatting

We will look at some of the auditing that I did (contained in external .py scripts) and discuss ideas about the data set below.

Information following is an initial python-scripted audit of the full OSM file.

In [1]:
```
#Importing useful python elements to examine data.
import xml.etree.ElementTree as ET
import pprint
from collections import defaultdict
import re
import csv
import codecs
import cerberus
import sqlite3
import os

OSM_FILE = "BeaverCounty.osm" # My OSM file of Beaver County, PA
SAMPLE_FILE = "BeaverCountySample.osm" # My sample file to be created below

k = 10 # Take every 10th element from the top level tags.

def get_element(osm_file, tags=('node', 'way', 'relation')):
    context = iter(ET.iterparse(osm_file, events=('start', 'end')))
    _, root = next(context)
    for event, elem in context:
        if event == 'end' and elem.tag in tags:
            yield elem
            root.clear()

with open(SAMPLE_FILE, 'wb') as output:
    output.write('<?xml version="1.0" encoding="UTF-8"?>\n'.encode())
    output.write('<osm>\n'.encode())

    # Write every kth top level element
    for i, element in enumerate(get_element(OSM_FILE)):
        if i % k == 0:
            output.write(ET.tostring(element, encoding='utf-8'))

    output.write('</osm>'.encode())
```

In [2]:
```
# Get the file size
os.stat('BeaverCounty.osm')
```

Out[2]:
```
os.stat_result(st_mode=33206, st_ino=844424930873890, st_dev=2523738793, st_nlink=1, st_uid=0, st_gid=0, st_size=105381269, st_atime=1603236179, st_mtime=1602340156, st_ctime=1601126629)
```

File Statistics:

File is 105381578 bytes, or 105.381578 Megabytes (st_size=105381578)

We will count elements, get unique users and list them and compare some of these findings with our SQL later:

In [3]:
```
# Count the number of each type of tag in main OSM file

def count_tags(filename):
    tree=ET.iterparse(filename)
    tags={}
    for event,elem in tree:
        if elem.tag not in tags.keys():
            tags[elem.tag]=1
        else:
            tags[elem.tag] = tags[elem.tag]+1
    return tags

with open(OSM_FILE,'rb') as f:
    tags=count_tags(OSM_FILE)
    pprint.pprint(tags)
f.close()
```

```
{'member': 25791,
 'nd': 551791,
 'node': 478084,
 'osm': 1,
 'relation': 940,
 'tag': 181551,
 'way': 53020}
```

In [4]:
```
# Find the number of users who have updated Beaver County, and list them. I omitted the list for final submission.

def get_users(element):
    return element.get('user')


def process_users(filename):
    users = set()
    for _, element in ET.iterparse(filename):
        if element.get('user'):
            users.add(get_users(element))
        element.clear()
    return users


with open(OSM_FILE,'rb') as f:
    users = process_users(OSM_FILE)

print(len(users))
#pprint.pprint(users)
f.close()
```

```
814
```

In [7]:
```
# Check formatting of the K attribute in the tags in the OSM file, code referenced from Udacity & Github
#

lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')


def key_type(element, keys):
    if element.tag == "tag":
        if lower.search(element.attrib['k']):
            keys['lower'] += 1
        elif lower_colon.search(element.attrib['k']):
            keys['lower_colon'] += 1
        elif problemchars.search(element.attrib['k']):
            keys['problemchars'] = keys['problemchars'] + 1
        else:
            keys['other'] += 1
            #print(element.attrib['k'])
            #print(element.attrib['v'])
    return keys


def process_keys_map(filename):
    keys = {"lower": 0, "lower_colon": 0, "problemchars": 0, "other": 0}
    for _, element in ET.iterparse(filename):
        keys = key_type(element, keys)

    return keys

with open(OSM_FILE,'rb') as f:
    keys = process_keys_map(OSM_FILE)
    pprint.pprint(keys)
f.close()
```

```
{'lower': 116841, 'lower_colon': 61630, 'other': 3080, 'problemchars': 0}
```

In [8]:

```
# Finding unique k (tag attrib['k']) and count

def unique_keys(filename):
    distinct_keys=[]
    count=1
    #Loop and count unique elements
    EL=get_element(filename, tags=('node', 'way', 'relation'))
    for element in EL:
        if element.tag=='node' or element.tag=='way':
            for tag in element.iter('tag'):
                if tag.attrib['k'] not in distinct_keys:
                    distinct_keys.append(tag.attrib['k'])
                    count+=1
    distinct_keys.sort()
    print("Total of unique keys is {}".format(count))

    return distinct_keys

    #pprint.pprint(distinct_keys)


unique_keys(OSM_FILE)  # Using real file as input to audit the addr:street key
```

Total of unique keys is 430

Out[8]: ['FIXME',
'NHS',
'abandoned:amenity',
'abandoned:railway',
'access',
'addr:city',
'addr:country',
'addr:housename',
'addr:housenumber',
'addr:postcode',
'addr:state',
'addr:street',
'addr:unit',
'admin_level',
'aerodrome:type',
'aeroway',
'air_conditioning',
'airmark',
'alt_name',
'amenity',
'amenity_1',
'area',
'atm',
'backrest',
'bar',
'barrier',
'basin',
'beacon:code',
'beacon:frequency',
'beacon:type',
'bicycle',
'bicycle_parking',
'board_type',
'boat',
'boundary',
'brand',
'brand:wikidata',
'brand:wikipedia',
'brewery',
'bridge',
'bridge:name',
'building',
'building:levels',
'building:levels:underground',
'building:material',
'building:part',
'building_1',
'bus',
'cables',
'capacity',
'capacity:disabled',
'capacity:parent',
'capacity:women',
'category',
'census:population',
'centre_turn_lane',
'communication:mobile_phone',
'construction',
'contact:phone',
'contact:website',
'content',
'covered',
'craft',
'created_by',
'crossing',
'crossing:barrier',
'crossing:bell',
'crossing:island',
'crossing:light',
'crossing_ref',
'cuisine',
'cycleway',
'cycleway:left',
'deep_draft',
'delivery',
'denomination',
'departures_board',
'description',
'designation',
'destination',
'destination:lanes',
'destination:ref',
'destination:ref:backward',
'destination:ref:lanes',
'destination:ref:lanes:backward',
'destination:ref:lanes:forward',
'destination:ref:to',
'destination:ref:to:forward',
'destination:ref:to:lanes',
'destination:ref:to:lanes:backward',
'destination:ref:to:lanes:forward',
'destination:street:lanes',
'destination:symbol',
'direction',
'dispensing',

```
'disused:leisure',
'disused:name',
'drive_in',
'drive_through',
'ele',
'electrified',
'email',
'emergency',
'end_date',
'entrance',
'expressway',
'faa',
'fax',
'fee',
'fence_type',
'fireplace',
'fixme',
'flashing_lights',
'floating',
'foot',
'footway',
'ford',
'frequency',
'fuel',
'fuel:biodiesel',
'fuel:biogas',
'fuel:cng',
'fuel:diesel',
'fuel:e10',
'fuel:e85',
'fuel:octane_95',
'fuel:octane_98',
'gauge',
'generator:method',
'generator:output:electricity',
'generator:source',
'generator:type',
'gnis:Class',
'gnis:County',
'gnis:County_num',
'gnis:ST_alpha',
'gnis:ST_num',
'gnis:county_id',
'gnis:county_name',
'gnis:created',
'gnis:feature_id',
'gnis:feature_type',
'gnis:id',
'gnis:import_uuid',
'gnis:reviewed',
'gnis:state_id',
'golf',
'golf_cart',
'guide_type',
'healthcare',
'healthcare:speciality',
'height',
'heritage',
'heritage:operator',
'hgv',
'hgv:lanes',
'highway',
'historic',
'hoops',
'horse',
'iata',
'icao',
'import_uuid',
'incline',
'indoor',
'industrial',
'information',
'inscription',
'intermittent',
'internet_access',
'is_in',
'junction',
'junction:ref',
'landuse',
'lanes',
'lanes:backward',
'lanes:both_ways',
'lanes:forward',
'layer',
'lcn',
'leaf_cycle',
'leisure',
'length',
'level',
'lit',
'loc_name',
'local_ref',
'location',
'lock',
'lock_name',
'man_made',
```

```
'manhole',
'material',
'maxheight',
'maxspeed',
'maxspeed:advisory',
'maxspeed:advisory:backward',
'maxspeed:advisory:forward',
'maxspeed:backward',
'maxspeed:forward',
'maxspeed:hgv',
'maxspeed:hgv:forward',
'maxspeed:source',
'maxstay',
'memorial',
'memorial:type',
'military',
'monitoring_station',
'mooring',
'motor_vehicle',
'mtb:scale',
'mtb:scale:uphill',
'name',
'name:backward',
'name:en',
'name:forward',
'name:ru',
'name_1',
'name_2',
'natural',
'navigationaid',
'network',
'network:wikidata',
'noexit',
'noname',
'noref',
'note',
'note:lanes',
'note:old_railway_operator',
'nudism',
'odbl',
'odbl:note',
'office',
'official_name',
'old_name',
'old_railway_operator',
'old_ref',
'old_ref:legislative',
'old_ref_legislative',
'oneway',
'oneway:bicycle',
'oneway:foot',
'open_air',
'opened',
'opening_hours',
'opening_hours:emergency',
'opening_hours:rubbish',
'operator',
'operator:type',
'operator:wikidata',
'operator:wikipedia',
'outdoor_seating',
'ownership',
'par',
'park_ride',
'parking',
'parking:lane:both',
'parking:lane:left',
'parking:lane:right',
'passenger_lines',
'payment:cash',
'payment:credit_cards',
'payment:debit_cards',
'payment:e_zpass',
'payment:license_plate',
'payment:mastercard',
'payment:visa',
'phone',
'phone_1',
'pipeline',
'place',
'plant:output:electricity',
'plant:source',
'playground',
'population',
'power',
'protect_class',
'protection_title',
'psv',
'public_transport',
'radar',
'railway',
'railway:position',
'railway:signal:direction',
'railway:signal:position',
'railway:switch',
'railway:track_ref',
```

```
'railway:turnout_side',
'ramp',
'razed:building',
'rcn_ref',
'recycling:cans',
'recycling:glass',
'recycling:plastic_bottles',
'recycling_type',
'ref',
'ref:isil',
'ref:left',
'ref:nrhp',
'ref:penndot',
'ref:right',
'ref:walmart',
'religion',
'removed:bridge',
'removed:highway',
'reservation',
'residential',
'roof',
'roof:levels',
'route',
'screen',
'second_hand',
'self_service',
'service',
'shelter',
'shelter_type',
'ship',
'shop',
'short_name',
'sidewalk',
'smoking',
'social_facility',
'social_facility:for',
'socket:chademo',
'socket:type1',
'source',
'source:access',
'source:addr',
'source:bicycle',
'source:bridge',
'source:building',
'source:construction',
'source:deep_draft',
'source:description',
'source:geometry',
'source:height',
'source:hgv',
'source:maxheight',
'source:maxspeed',
'source:motor_vehicle',
'source:name',
'source:oneway',
'source:pkey',
'source:ref',
'source:ref:penndot',
'source:shop',
'source:start_date',
'source:tunnel',
'source_opened',
'source_ref',
'sport',
'sport_1',
'start_date',
'stop',
'studio',
'substance',
'supervised',
'surface',
'surveillance:type',
'tactile_paving',
'takeaway',
'tidal',
'tiger:cfcc',
'tiger:county',
'tiger:name_base',
'tiger:name_base_1',
'tiger:name_base_2',
'tiger:name_direction_prefix',
'tiger:name_direction_prefix_1',
'tiger:name_direction_suffix',
'tiger:name_type',
'tiger:name_type_1',
'tiger:name_type_2',
'tiger:reviewed',
'tiger:separated',
'tiger:source',
'tiger:tlid',
'tiger:upload_uuid',
'tiger:zip',
'tiger:zip_left',
'tiger:zip_left_1',
'tiger:zip_left_2',
'tiger:zip_left_3',
```

```
'tiger:zip_left_4',
'tiger:zip_right',
'tiger:zip_right_1',
'tiger:zip_right_2',
'tiger:zip_right_3',
'tiger:zip_right_4',
'tiger:zip_right_6',
'toilets:wheelchair',
'toll',
'tourism',
'tower:construction',
'tower:type',
'tracktype',
'traffic_signals',
'traffic_signals:direction',
'trail_visibility',
'tram',
'tunnel',
'turn',
'turn:backward',
'turn:forward',
'turn:lanes',
'turn:lanes:backward',
'turn:lanes:both_ways',
'turn:lanes:forward',
'type',
'unsigned',
'unsigned_ref',
'url',
'usage',
'vehicle',
'vending',
'voltage',
'voltage-high',
'wall',
'water',
'water_source',
'waterway',
'website',
'wetland',
'wheelchair',
'width',
'wifi',
'wikidata',
'wikipedia',
'wires']
```

In [9]:

```
#Looking to see what types of things people have entered in my county sample.

def unique_keys(filename):
    distinct_keys=[]
    count=1

    EL=get_element(filename, tags=('node', 'way', 'relation'))
    for element in EL:
        if element.tag=='node' or element.tag=='way':
            for tag in element.iter('tag'):
                if tag.attrib['k'] not in distinct_keys:
                    distinct_keys.append(tag.attrib['k'])
                    count+=1
    distinct_keys.sort()
    print("There are {} different types of keys in my file.".format(count))

    return distinct_keys

    #pprint.pprint(distinct_keys)


unique_keys(SAMPLE_FILE)  # Using Sample file as input to audit the addr:street key
```

There are 250 different types of keys in my file.

Out[9]: ['NHS',
'abandoned:railway',
'access',
'addr:city',
'addr:housenumber',
'addr:postcode',
'addr:state',
'addr:street',
'admin_level',
'aeroway',
'alt_name',
'amenity',
'area',
'backrest',
'barrier',
'bicycle',
'boat',
'boundary',
'brand',
'brand:wikidata',
'brand:wikipedia',
'bridge',
'bridge:name',
'building',
'building:levels',
'building:levels:underground',
'building:material',
'building:part',
'building_1',
'bus',
'cables',
'capacity',
'census:population',
'communication:mobile_phone',
'content',
'covered',
'craft',
'created_by',
'crossing',
'crossing:barrier',
'crossing:bell',
'crossing:light',
'cuisine',
'cycleway',
'deep_draft',
'denomination',
'departures_board',
'description',
'designation',
'destination',
'destination:ref',
'destination:ref:backward',
'destination:ref:lanes:forward',
'destination:ref:to',
'destination:ref:to:forward',
'destination:ref:to:lanes:forward',
'destination:street:lanes',
'direction',
'disused:leisure',
'disused:name',
'drive_through',
'ele',
'electrified',
'emergency',
'end_date',
'entrance',
'expressway',
'fax',
'fee',
'fence_type',
'fixme',
'foot',
'footway',
'frequency',
'gauge',
'generator:method',
'generator:output:electricity',
'generator:source',
'generator:type',
'gnis:Class',
'gnis:County',
'gnis:County_num',
'gnis:ST_alpha',
'gnis:ST_num',
'gnis:county_id',
'gnis:county_name',
'gnis:created',
'gnis:feature_id',
'gnis:feature_type',
'gnis:id',
'gnis:import_uuid',
'gnis:reviewed',
'gnis:state_id',
'golf',
'golf_cart',

```
'guide_type',
'healthcare',
'healthcare:speciality',
'hgv',
'highway',
'historic',
'horse',
'iata',
'icao',
'import_uuid',
'incline',
'indoor',
'information',
'is_in',
'junction',
'junction:ref',
'landuse',
'lanes',
'lanes:backward',
'lanes:both_ways',
'lanes:forward',
'layer',
'lcn',
'leisure',
'lit',
'location',
'lock',
'lock_name',
'man_made',
'material',
'maxheight',
'maxspeed',
'maxspeed:advisory',
'maxspeed:hgv',
'memorial',
'mooring',
'motor_vehicle',
'name',
'name_1',
'natural',
'navigationaid',
'network',
'noexit',
'noref',
'note',
'note:old_railway_operator',
'odbl',
'office',
'official_name',
'old_name',
'old_railway_operator',
'old_ref',
'old_ref:legislative',
'old_ref_legislative',
'oneway',
'opened',
'opening_hours',
'operator',
'operator:type',
'par',
'park_ride',
'parking',
'parking:lane:both',
'parking:lane:left',
'parking:lane:right',
'passenger_lines',
'payment:cash',
'payment:credit_cards',
'payment:debit_cards',
'payment:e_zpass',
'payment:license_plate',
'payment:mastercard',
'payment:visa',
'phone',
'place',
'playground',
'population',
'power',
'public_transport',
'railway',
'railway:turnout_side',
'rcn_ref',
'ref',
'ref:penndot',
'religion',
'roof:levels',
'second_hand',
'service',
'shelter',
'shelter_type',
'ship',
'shop',
'sidewalk',
'social_facility',
'social_facility:for',
'source',
```

```
'source:bridge',
'source:building',
'source:deep_draft',
'source:geometry',
'source:hgv',
'source:name',
'source:ref',
'source:ref:penndot',
'source_opened',
'sport',
'stop',
'substance',
'surface',
'surveillance:type',
'takeaway',
'tiger:cfcc',
'tiger:county',
'tiger:name_base',
'tiger:name_base_1',
'tiger:name_direction_prefix',
'tiger:name_direction_suffix',
'tiger:name_type',
'tiger:name_type_1',
'tiger:reviewed',
'tiger:separated',
'tiger:source',
'tiger:tlid',
'tiger:upload_uuid',
'tiger:zip_left',
'tiger:zip_left_1',
'tiger:zip_left_2',
'tiger:zip_left_3',
'tiger:zip_right',
'tiger:zip_right_1',
'tiger:zip_right_3',
'tiger:zip_right_6',
'toilets:wheelchair',
'toll',
'tourism',
'tower:type',
'tracktype',
'traffic_signals:direction',
'tunnel',
'turn',
'turn:lanes',
'turn:lanes:backward',
'turn:lanes:forward',
'usage',
'voltage',
'wall',
'water',
'waterway',
'website',
'wheelchair',
'width',
'wikidata',
'wikipedia',
'wires']
```

Now that we've explored the data in Python...

Let's take another look at the database created by the python scripts and SQLITE code. I will browse the county and the data entered for it to see what's out there, and by whom things may have been added.

How many restaurants are listed, and what type?

- Here is the SQL I ran:

```
select count(*) as Quantity, value from ways_tags where key="cuisine"
group by ways_tags.VALUE
order by Quantity DESC;
```

- And my results were as follows.

Quantity	value
12	burger
11	american
3	italian
2	sandwich
2	pizza
2	ice_cream
1	wings
1	steak_house
1	donut
1	coffee_shop
1	chinese
1	chicken
1	asian;chinese
1	american;international;pizza
1	american;burger;ice_cream
1	Custard,_Ice_Cream,_Burgers,_Hot_Dogs,_Fries

Interestingly enough, I would speculate that the name of the place listed at the bottom, which has a sloppy format with underscores, would be a famous local taco/custard place called Hank's. I will verify this with another query.

Turns out I was wrong, and I learned of another similar place in the area. Here's how I found out:

First, I had to find the value associated with that entry in my cuisine query:

```
select * from ways_tags where key="cuisine" and value ="Custard,_Ice_Cream,_Burgers,_Hot_Dogs,_Fries";
```

Next I joined tables to get all the info for the ID I found:

```
select * from ways_tags inner join ways on ways_tags.id=ways.id where ways_tags.id="373434539";
```

I learned of a place called "Young's Custard", as seen below and updated as of December 11th, 2019:

value	category	id	user	uid	version	changeset	timestamp
restaurant	regular	373434539	skquinn	243003	4	78250214	2019-12-11T10:04:13Z
yes	regular	373434539	skquinn	243003	4	78250214	2019-12-11T10:04:13Z
Custard,_Ice_Cream,_Burgers,_Hot_Dogs,_Fries	regular	373434539	skquinn	243003	4	78250214	2019-12-11T10:04:13Z
no	regular	373434539	skquinn	243003	4	78250214	2019-12-11T10:04:13Z
Young's Custard	regular	373434539	skquinn	243003	4	78250214	2019-12-11T10:04:13Z
Wednesday - Monday Until 10:00 PM	opening_hours	373434539	skquinn	243003	4	78250214	2019-12-11T10:04:13Z

You can visit this page to learn more about them, if you are curious: https://www.facebook.com/youngscustard/ (https://www.facebook.com/youngscustard/)

Let's analyze other interesting data

Number of unique users:

```
select count(distinct(allusers.uid))
from (select uid from nodes union all select uid from ways) allusers;
```

Result: 807

*Note - this is just shy of what Python gathered before, which was 814.

Top 25 users by number of contributions:

```
SELECT user, COUNT(allusers.user) as Quantity
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) allusers
group by user
order by Quantity desc limit 25;
```

user	Quantity
balcoath	134166
GeoKitten_import	66442
GeoKitten	33596
DonovanG	32258
BMBurgh	18759
woodpeck_fixbot	8565
OSchlüter	8376
jmpaxton	8220
Omnific	8107
czwalga	7866
Claytonlukes77	7831
Roadsguy	7790
rickmastfan67	6849
abbafei	6424
Jeff Bearer	6237
jonwit_import	5908
JJD17	5768
AndrewSnow	5352
andrewrpucci	5303
dchiles	4605
Uniplex	4177
freebeer	3551
jleedev	3424
Scharf Energy Systems	3153
NE2	3077

How about amenities...what is there in Beaver County, PA?

```
select value, count(*) as quantity from nodes_tags where key='amenity'
group by value
order by quantity desc limit 20;
```

Here's what I found:

value	quantity
school	88
place_of_worship	80
restaurant	45
grave_yard	35
fast_food	31
parking	29
bank	26
police	17
cafe	16
post_office	15
bench	15
waste_basket	12
kindergarten	12
fuel	12
fire_station	12
bar	12
townhall	11
pharmacy	10
library	10
doctors	9

> It's noteworthy that there are 21 zip codes, but only 15 post offices listed. Per this site, there are 19 total post offices in the county: https://www.postallocations.com/pa/county/beaver (https://www.postallocations.com/pa/county/beaver)

Let's have a look at how zip codes turned out:

I ran this SQL. You will notice I accounted for an issue with "Tiger GPS" data which I found in my exploration, where they have zip codes labeled as fields other than "postcode":

```
select value, count(value) as quantity from ways_tags where key in ("postcode", "zip_left", "zip_right")
group by value
order by quantity DESC;
```

And I found this data:

value	quantity

15027| 118 15050| 94 16141| 77 15056| 77 15126| 75 15081| 45 16157| 24 16120| 22 16063| 17 16136| 12 15077| 11 16037| 4 15065| 4 15057| 4 15064| 2 15063| 2 15026:15126| 2 16115:16141| 1 15231| 1 15101| 1|

To further analyze the data, I decided to see how many zip codes I could find that were not part of Beaver County, per the beavercounty.org website:

```
select distinct value from ways_tags where key in ("postcode","zip_left","zip_right") and value not in ("15001","15003","15005","15009","15010",
"15026","15027","16115","16123","15042",
"15043","15050","16136","15059","15061",
"16141","15066","15052","15074","15077","15081")
order by value;
```

These are the zip codes not included in Beaver County, per their website. One exception among possible others is the mailing addresses with zip code "15026" which are in Beaver County, despite this being an Allegheny County zip code:

> 15026:15126 15046 15056 15057 15063 15064 15065 15101 15108 15126 15143 15231 16037 16063 16115:16141 16117 16120 16157

Other ideas about the dataset

The data set could be improved by adding attributes to each node which correlate to the questions asked on Google Maps, of the Google Guides.

For example, I regularly answer questions about wheelchair accessibility, take-out, all-you-can-eat options, delivery, various specific dishes served at a restaurant, products sold at a store, and more. If high-level general data from Google Guides answers could be imported to OpenStreetMap as additional tag/key values, the available data on openstreetmap as well as the potential data to be queried in future analysis would be exponentially expanded.

This would require a partnership between Google and OpenStreetMap, and I will not speculate as to the actual complications, but I would guess this would not be an easy relationship to establish, given that Google has a corner on this market today. If they wanted to, OpenStreetMap could implement questions to be answered about various entries on the map for restaurants, businesses, etc.

Additionally, it would be great if you could easily pull and update "Tiger GPS data" on the site in a similar fashion. If I could sit and click through Tiger data and update it and submit the change, much of the data on the site could be easily read and more consistent for analysis. For example, I looked to see what kind of "Tiger" data was in my ways_tags table, and here's what I found:

```
select distinct key from (select * from ways_tags where ways_tags.category="tiger");
```

key
source
tlid
separated
county
name_base
name_type
zip_left
zip_right
name_base_1
name_direction_suffix
name_type_1
zip_left_1
zip_right_1
zip_left_2
zip_left_3
zip_right_3
name_direction_prefix
zip_right_2
zip_left_4
upload_uuid
name_base_2
zip_right_6
name_type_2
name_direction_prefix_1
zip

This small sample shows that even with an item such as "postcode", there are a number of variations on the data (known locally as a zip code) which make it tough to find the full sample.

The difficulty in fixing this problem could lie in limiting users to the types of attributes they can enter, without knowing all the potential types and values that are valid in various regions.

Conclusions

It is evident that allowing users to free-form data causes some interesting results to pop up. There are a number of issues also caused by the tiger GPS data, which spreads the values for an item such as a "Zip", or "postcode" in this case, across multiple key values.

I was shocked at the number of users who have contributed, and the amount they contributed, given the rural area that is covered in my analysis and the fact that before this project I'd never heard of openstreetmap.org (https://www.openstreetmap.org) myself.

I think there are 2 ways the site could obtain cleaner data more efficeintly.

1. Having a process to submit a new item, such as "cuisine type" which would allow for no less than 3 other users to validate or edit before it's allowed to be used would prevent items from making it to prime time, such as "Custard,_Ice_Cream,_Burgers,_Hot_Dogs,_Fries" which I found in my data set.
2. Having an algorithm which validates data upon entry, based on country or region could prevent some erroneous entries in more standardized fields such as phone numbers and zip codes.

The problem with either of these is a level of effort, or for the peer review approval process, it may cause some data to never make it to the site.

Additionally, there is likely a lot of opportunity to add data to the map. It's highly unlikely that all of the restaurants in the county are accounted for in the short list above.

References

1. Github
2. stackoverflow
3. Udacity course materials