OpenStreetMap Project
Data Wrangling with MongoDB

Blake Justice

Map Area: Los Angeles, California

Problems Encountered in the Map

# Inconsistent Zip Codes

- The zip codes in my dataset do not have consistent formatting. For example, the following zip codes were found in my dataset and possibly represent the same areas.
    - 92656-2601
    - 92656
- The second zip code (called a ZIP+4 zipcode) is more specific than the typical 5-digit zip code. To clean this, if a zip code included 9 characters instead of 5, I stripped the last 4 digits off the zip code. Below is the code I used to do this.

```
long_zip = db.osm_v2.find({'address.postcode':{'$regex':'^\d.*-'}})
for x in long_zip:
      x['address']['postcode']=
      re.sub('-.*','',str(x['address']['postcode']))
      db.osm_v2.save(x)
```

# Inconsistent Cities

- After inserting the data into MongoDB, I found that there were 296 different cities listed in my collection. I wasn't expecting this since my OSM data was only supposed to be for Los Angeles, but initially thought that it may be broken down by neighborhood. Below is the pipeline to find the number of unique cities in my dataset.

```
db.osm_v2.aggregate([{"$match":{"address.city":{"$exists":1}}},
{"$group":{"_id":"$address.city","count":{"$sum":1}}},{"$group":{"_id":1,
"count":{"$sum":1}}},{"$sort":{"count":1}}])
```

- Three of the cities values I found were Los Angeles, San Diego, and Beverly Hills. San Diego is a completely different city, so this was unexpected. Beverly Hills is a neighborhood in Los Angeles so that made more sense to include in this dataset. To back up my thought about specific neighborhoods being labeled as cities, I checked the following San Diego address to see if the neighborhood was in fact San Diego. Turns out it was not. The address is actually in Palomar Mountain, which leads me to believe that the other city fields outside of the document below may be incorrect as well.

```
{'is_in:country_code': 'US', 'is_in:country': 'United States of America',
'source': 'SanGIS Addresses Public Domain (http://www.sangis.org/)',
'is_in:state_code': 'CA', 'id': '595965284', 'address': {'housenumber':
'22220', 'street': 'Crestline Rd', 'city': 'San Diego', 'postcode':
'92060', 'country': 'US'}, 'created': {'changeset': '3405180',
'timestamp': '2009-12-19T07:52:42Z', 'version': '1', 'uid': '17490',
'user': 'Adam Geitgey'}, 'pos': [33.311308, -116.850192], '_id':
ObjectId('56246935ee4011075d823b79'), 'is_in:city': 'San Diego', 'type':
'node', 'is_in:state': 'California'}
```

- I believe this dataset would be better named the “Southern California Metropolitan Area” because it includes areas outside of Los Angeles. This would help avoid confusing other users when they come across these problems in the dataset.

# Inconsistent House Numbers

- There are 101 house numbers that include non-numeric characters. I discovered this using the following query.

```
db.osm_v2.aggregate(
[{"$match":{"address.housenumber":{"$regex":"[A-Za-z]"}}},
{"$group":{"_id":"$address.housenumber","count":{"$sum":1}}},{"$group":{"
_id":1,"count":{"$sum":1}}},{"$sort":{"count":1}}])
{'count': 101, '_id': 1}
```

- Three of these have been provided below.
  - 2660 Park Center Drive
  - 2401 West Alameda Avenue
  - 39252 Winchester Rd Murrieta, CA 92563

Ideally, this field would only include the house number, not the entire street address.

# Inconsistent Street Names

- Some of the documents in my collection have abbreviated street types. For example, St. and Street are the same, but could cause data inconsistencies when querying the database. To fix this, I used the following mapping to update a section of the affected documents.

```
short_streets = db.osm_v2.aggregate(
[{"$match":{"address.street":{"$regex":"[A-Za-z]\."}}}])

street_endings= {"Blvd.":"Boulevard ","St.":"Street ","Ave.":"Avenue
","Dr.":"Drive ","Ctr.":"Center ","Rd.":"Road ",
"Blvd ":"Boulevard ","St ":"Street ","Ave ":"Avenue ","Dr ":"Drive ","Ctr
":"Center ","Rd ":"Road "}

for x in short_streets:
    street = x['address']['street']
    for k,v in street_endings.items():
        if k in street:
            fixed_street = re.sub(k,v,str(street)).strip()
            x['address']['street'] = fixed_street
            db.osm_v2.save(x)
```

Data Overview

- This section contains high-level statistics about my dataset named *los_angeles-california.osm*.The sample file for grading is named *sample_final.osm***.**

#File Size

```
los_angeles-california.osm: 1.22 GB
sample_final.osm 12.4 MB
```

#Number of Documents in db.project3.osm_v2 MongoDB Collection

```
db.osm_v2.count()
5,533,000
```

#Number of Tag Types

```
db.osm_v2.find({"type":"node"}).count()
5,268,722
db.osm_v2.find({"type":"way"}).count()
263,722
```

#Number of Unique Users

```
db.osm_v2.aggregate([{"$group":{"_id":"$created.user","count":{"$sum":1}}
},{"$group":{"_id":1,"count":{"$sum":1}}},{"$sort":{"count":-1}}])
2,790
```

#User with Most Contributions

```
db.osm_v2.aggregate([{"$group":{"_id":"$created.user","count":{"$sum":1}}
},{"$sort":{"count":-1}},{"$limit":1}])
{'count': 546384, '_id': 'woodpeck_fixbot'}
```

#Top 3 Cities Included in Data

```
db.osm_v2.aggregate([{'$match':{"address.city":{"$exists":1}}},
{"$group":{"_id":"$address.city","count":{"$sum":1}}},{'$sort':{"count":-
1}},{"$limit":3}])
{'count': 14114, '_id': 'San Diego'}
{'count': 12217, '_id': 'Lake Forest'}
{'count': 11252, '_id': 'Irvine'}
```

Additional Data Exploration and Other Ideas

- After analyzing the entries with amenities data, I found some interesting insights into the area's restaurant scene. For example, initially I expected McDonalds to be the most common fast food chain in the area due to it's popularity, but it turns out that Subway is more common. This may only be because the dataset is incomplete.

  Another interesting insight I found is that places of worship appear to be more common than parking areas, restaurants, and fast food amenities. If the dataset were more complete, I would do a deeper dive into amenities and their correlations to see if this could point to any information about the population's demographics. This could be used to predict what other types of amenities not listed would also be common in the area. Some example queries have been provided below.

#Top 5 Amenities

```
db.osm_v2.aggregate([{'$match':{"amenity":{'$exists':1}}},
{'$group':{'_id':'$amenity','count':{'$sum':1}}},{'$sort':{'count':-1}},{
'$limit':5}])
{'count': 3892, '_id': 'place_of_worship'}
{'count': 3806, '_id': 'school'}
{'count': 2271, '_id': 'parking'}
{'count': 1873, '_id': 'restaurant'}
{'count': 1341, '_id': 'fast_food'}
```

#Top 5 Restaurant Cuisines

```
db.osm_v2.aggregate([{'$match':{'amenity':'restaurant'}},
{'$group':{'_id':'$cuisine','count':{'$sum':1}}},
{'$sort':{'count':-1}},{'$limit':5}])
 {'count': 795, '_id': None}
 {'count': 173, '_id': 'american'}
 {'count': 152, '_id': 'mexican'}
 {'count': 87, '_id': 'pizza'}
 {'count': 73, '_id': 'italian'}
```

#Top 3 Fast-Food Restaurants

```
db.osm_v2.aggregate([{'$match':{'amenity':'fast_food'}},
{'$group':{'_id':'$name','count':{'$sum':1}}},
{'$sort':{'count':-1}},{'$limit':3}])
{'count': 117, '_id': 'Subway'}
{'count': 113, '_id': "McDonald's"}
{'count': 67, '_id': 'Jack in the Box'}
```

- In conclusion, I believe this dataset could be improved in a few ways. The main improvement I see as being most beneficial is checking for entry completion. Currently, most of the entries don't include address or amenity data. If there was a requirement to enter these, the dataset would be more complete and provide a more accurate representation of the area.The benefit of having this is a more complete dataset as mentioned above. Also, further insights could possibly be gained from the dataset and lead to more accurate predictions and theories about the geographic area.

  However, having a requirement for data entry may cause users to simply not submit entries altogether if they are not inclined to include this data in the first place. Another issue I could see occurring is that some users may just not have this type of data at their disposal, and as a result, hinder them from submitting any data altogether.

- Another improvement would be cleaner data. As I stated in section 1, the entries that included address information were at times inaccurate. If there were some type of cleaning mechanism or bot between the submission process and the data being uploaded to OSM, this would prove beneficial to the analysis process. This would be beneficial because it could prevent incorrect types of data being submitted to certain fields, or clean them after being submitted, which would result in a cleaner and more accurate dataset.

  However, having a cleaning mechanism such as a bot would require resources to create and maintain overtime. This would most likely cause the OSM team to incur additional financial and resourcing costs that are currently non-existent. Another possible issue is creating a faulty cleaning mechanism, which could do more harm than good when cleaning data. For example, a cleaning bot could update incorrect fields, flag errors that were actually correct, and so on.

External Sources:

https://docs.python.org/2/library/xml.etree.elementtree.html
https://docs.mongodb.org/getting-started/python/
https://docs.mongodb.org/getting-started/python/aggregation/
https://docs.mongodb.org/manual/reference/operator/aggregation/match/
https://docs.mongodb.org/manual/reference/method/db.collection.update/