OpenStreetMap Sample Project - Data Wrangling with MongoDB

Melissa Bai

Table of Content

1. Problems Encountered

After downloading a small size file around Palo Alao area and auditing the street names. There are mainly two problems encountered in the dataset

- Over abbreviated street names
- Different naming conventions for zip code

Over abbreviated street names

Street names have inconsistent suffix, such as ‘Ave’, ‘Dr.’, ‘Rd’, ‘St’ instead of complete names.

Examples:

'Ave': set(['California Ave', 'Forest Ave', 'S California Ave']),
'Dr.': set(['Campus Dr.']),

'Rd': set(['Embarcadero Rd']),
'St': set(['Ramona St']),

Code Snippets:

```
def audit_street_name(osm_file):
    street_types = defaultdict(set)
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    osm_file.close()
    return street_types
```

For these abbreviated street suffix, I have normalized them into full street names.
The 'update_name' function and 'better_name' function update the dataset with complete street name suffix.

```
def update_name(name, mapping):
    name_list = name.split(" ")
    suffix = name_list[-1]
    if suffix in mapping:
        new_suffix = mapping[suffix]
        name_list[-1] = new_suffix
    name = " ".join(name_list)
    return name

def better_name():
    st_types = audit_street_name(osm_file)

    for st_type, ways in st_types.iteritems():
        for name in ways:
            better_name = update_name(name, mapping)
            name = better_name
            print name, "=>", better_name
```

=>
Embarcadero Rd => Embarcadero Road
Campus Dr => Campus Drive
Ramona St => Ramona Street
Santa Cruz avenue => Santa Cruz Avenue

Zip Code has different format

Zip code in Palo Alto are relatively clean, with slightly different formats. Some only contain five digits, some with additional 4 digit, only one with two different zip codes ('94305;94309') separated by comma.

Examples:

```
{'normal': set(['94025',
                '94036',
                '94301',
                '94301-2019',
                '94303',
                '94304',
                '94304-1050',
                '94305',
                '94305-6015',
                '94305;94309',
                '94306'])}
```

Code Snippets:

```
zipcode = re.compile(r'\d{5}(?:-\d{4})?')

def audit_zip_type(zip_types, postal):
    z = zipcode.search(postal)
    if z:
        zip_types['normal'].add(postal)
    else:
        zip_types['non_normal'].add(postal)

def is_zip(elem):
    return (elem.attrib['k'] == "addr:postcode")

def audit_zipcode_name():
    zip_types = defaultdict(set)
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_zip(tag):
                    audit_zip_type(zip_types, tag.attrib['v'])
    pprint.pprint(dict(zip_types))
```

2. Overview of the Data

File Sizes:

OpenStreetMap.osm 52.5MB
OpenStreetMap.osm.json 58.2MB

Number of documents

```
> db.pa.find().count()
258925
```

Number of nodes

```
> db.pa.find({"type":"node"}).count()
233324
```

Number of ways

```
> db.pa.find({"type":"way"}).count()
25601
```

Number of unique users

```
> db.pa.distinct({"created.user"}).length
314
```

Top 1 contributing user

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

[{u'_id': u'oldtopos', u'count': 55320}]
```

3. Other ideas about the datasets

The following code will be used to explore the dataset. Only the pipeline list will be modified for different exploration.

Further Exploration

```
from pprint import pprint

def get_db(db_name):
    from pymongo import MongoClient
    client = MongoClient('localhost:27017')
    db = client[db_name]
    return db

def make_pipeline():

    pipeline = [{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                {"$sort":{"count":-1}}, {"$limit":1}]


    return pipeline

def aggregate(db, pipeline):
    return db.pa.aggregate(pipeline)

def main():
    db = get_db('test')
    pipeline = make_pipeline()
    result = aggregate(db, pipeline)
    pprint(list(result))

main()
```

Top Amenities in Palo Alo

```
pipeline = [
            {"$group":{"_id":"$amenity", "count":{"$sum":1}}},
```

```
            {"$sort":{"count":-1}}
        ]

=>
[{u'_id': None, u'count': 257979},
 {u'_id': u'parking', u'count': 257},
 {u'_id': u'restaurant', u'count': 158},
 {u'_id': u'post_box', u'count': 69},
 {u'_id': u'fast_food', u'count': 42},
 {u'_id': u'school', u'count': 41},
 {u'_id': u'bench', u'count': 39},
 {u'_id': u'bicycle_parking', u'count': 36},
 {u'_id': u'cafe', u'count': 35},
 {u'_id': u'place_of_worship', u'count': 31},
...
```

Top cuision in Palo Alto area

```
 pipeline = [
            {"$group":{"_id":"$cuisine", "count":{"$sum":1}}},
            {"$sort":{"count":-1}}
        ]

=>
[{u'_id': None, u'count': 258791},
 {u'_id': u'mexican', u'count': 15},
 {u'_id': u'pizza', u'count': 12},
 {u'_id': u'italian', u'count': 12},
 {u'_id': u'coffee_shop', u'count': 8},
 {u'_id': u'french', u'count': 8},
 {u'_id': u'chinese', u'count': 7},
 {u'_id': u'sandwich', u'count': 7},
 {u'_id': u'thai', u'count': 6},
 {u'_id': u'middle eastern', u'count': 5},
 {u'_id': u'burger', u'count': 5},
...
```

Top religion in Palo Alto area

```
pipeline = [{"$match": {"amenity": "place_of_worship"} },
```

```
            {"$group":{"_id":"$religion", "count":{"$sum":1}}},
            {"$sort":{"count":-1}}]

=>
[{u'_id': u'christian', u'count': 28},
 {u'_id': u'jewish', u'count': 1},
 {u'_id': None, u'count': 1},
 {u'_id': u'yogic', u'count': 1}]
```

From the above queries we know that the most common amenties in palo alto area are parking lots, given it's a suburban area, it's not much of a surprise. The most common cuision is Mexican food, this also makes sense as California has a rich Mexican culture. And the most popular religion is Christian. A little bit surprised by how few religions are here, it also makes sense since may be silicon valley has less of a religious atmosphere than other places. But Christianity still remains as the top one.

Problems encountered

For the street suffix cleanup, I fixed the abbreviation at the end of the street. However, I found there are cases where street abbreviations happen in the middle of a street(i.e. Campus Dr. West, S California Ave). It would be great if further cleanup could be implemented to these cases as well, but we need to be carefull not to overwrite some actual names instead of abbreviations. Therefore, how to idenfiy patterns of street abbreviation vs actual names would be an important step to do here.

Inviting users to add notes to places

One of a Christian church I found from the database has a note: "Although it might allow for non-Christian services, we want it to render as a church on the map". I thought this is interesting and makes the places more detailed and fun. This is also helpful for people who are looking for say a church to check out. If OpenStreetMap can incentise users or place managers to add notes or even advertisement for their places, it will make the map more informative as well as higher data quality.

Summary

Overall I think Palo Alto area is well cleaned. The timestamp on the database ranges from 2008 to 2016, which shows that people are actively contributing to the map. This is really great.

While I would love to clean up a broader reagion in bay area, file size is a limiting factor for me to choose a bigger area. And also I did not use the geospatial query of MongoDB. In the future, I would love to pull data from Google Maps API and compare the difference with OpenStreet Map.