OpenStreetMap
Data Wrangling with MongoDB

Map Area: Boston, MA, United States
https://www.openstreetmap.org/relation/2315704
https://mapzen.com/data/metro-extracts/metro/boston_massachusetts/

1. Problems Encountered in the Map

After download the boston osm data, I run it against several python audit files, I noticed five main problems with the data, which I will discuss in the following order:

- Street name format is not uniform, some are full name, some are simplified name
- Inconsistent postal codes
- Inconsistent city name format
- Inconsistent state name format, and one mistake shows state of WA
- Inconsistent country name format

Inconsistent street name

By auditing the data using audit.py, I can see there is not a uniform format for street name, and the street name are shown in various format. For example, the last word of the "Street" in the street name have this following format:

"St,"
"St."
"Street."
"street"
"St"

There are many other similar cases. I wrote a function in audit.py to unify the last word appeared in the street name.

Inconsistent city name

By auditing the data using audit_city.py, most of the city name shown up in the correct format, with the first letter of the city name capitalized, and no state name is immediately following the city name.
For example, for Cambridge city alone, there are three formats: “Cambridge”, “Cambridge, MA”, “Cambridge, Massachusetts”. I wrote a function in the audit_city.py to update the city format, such that all “Cambridge, MA” and “Cambridge, Massachusetts” become Cambridge.

Inconsistent postal code

By auditing the data using audit_zip.py, I can see there are two major format of postcode, one is 5-digit zip code, the other is 5-digit plus 4-digit extension. The following regular expression in the function below include both of this format. The idea here is to find out those irregular ones for later updating.

```
def audit_zip(postcode):
    unexpected = []
    if not bool(re.match('^[0-9]{5}(?:-[0-9]{4})?$',postcode)):
        unexpected.append(postcode)
```

Among those unexpected ones, there are postcodes with mistakes. For example, there is one special case where the zip code is only 4 digit, 0239. Based on my knowledge of Boston area, it should be 02139, I wrote a function in the audit_zip.py by mapping to correct the zip code accordingly. And there are several cases, the postcode is not a number, simply state name "MA", in this case, I changed the "MA" to "N/A". I was hoping to format the zip code such that each zip code has the four digit extension, as I think with the extension, it will be easy to target the area for special jobs such as mail delivering. But without other resources available, it will be a daunting task.

Inconsistent state name format

By auditing the data using audit_state.py, I can see there are one major format of state name "MA", but there are various other kind of format, such as

"Ma",
"ma",
"MA- MASSACHUSETTS",
"MASSACHUSETTS",
"Massachusetts"

I wrote a function in audit_state.py to change the state into the uniform format "MA".

Inconsistent country name format

By auditing the data using audit_country.py, I can see there are two different major format of country name, one is "US", the other is "USA", since "US" has a greater percentage, I wrote a function in audit_country.py to change the country into the uniform format "US".

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

# File sizes

Boston_massachusetts.osm .... 436 MB
Boston_massachusetts.json .... 505 MB

# Number of documents loaded into mongo db

```
> db.boston.find().count()
2249763
```

# Number of nodes

```
> db.char.find({"type":"node"}).count()
1939475
```

# Number of ways

```
> db.char.find({"type":"way"}).count()
310288
```

# Number of unique users

```
> len(db.char.distinct("created.user"))
1385
```

Amenity distribution in different cities in boson area

Transportation

Boston transportation is interesting to people living here like me. So I first look at the places operated by MBTA. The query shows that there are 4783 spots.

```
>query = {"tags.operator": "MBTA"}
>print db.boston.find(query).count()
```

There are various subway lines and commuter rail lines in Boston area. I used a pipeline to see which subway line is the most popular one that the MBTA has most spots.

```
pipeline = [
   {
      "$group":{
         "_id": "$tags.line",
         "count":{"$sum":1}
         }
   },
   {
      "$sort":{
         "count":-1
         }
   },
]
db.boston.aggregate(pipeline)
[doc for doc in db.boston.aggregate(pipeline)]
```

```
[{u'_id': u'Green', u'count': 123},
 {u'_id': u'Red', u'count': 93},
 {u'_id': u'Blue', u'count': 84},
```

I can see the green line has the most MBTA spots count. The green line has 4 different sublines that connects the suburbs to the center of Boston city. While other lines do not have sublines.

Amenity-hospitals
The Boston area is made up of different cities(towns). The Boston city is famous for its health care. Which town has the largest number of hospitals?

```
pipeline = [
   {"$match":{"tags.amenity":"hospital","address.city":{"$exists":1} }},

   {"$group":{"_id":"$address.city",
           "count":{"$sum":1}
         }},
   {
      "$sort":{
         "count":-1
         }
   }
]

projection = {"adress.city":1}
db.boston.aggregate(pipeline)
[doc for doc in db.boston.aggregate(pipeline)][:5]
```

```
[{u'_id': u'Boston', u'count': 13},
 {u'_id': u'Jamaica Plain', u'count': 3},
 {u'_id': u'Brighton', u'count': 2},
 {u'_id': u'Brookline', u'count': 2},
 {u'_id': u'Cambridge', u'count': 2}]
```

It is not surprising that there are 13 hospitals in boson city, while in the neighboring cities, there are only 1-3 hospitals.

Amenity-restaurants
Which town has the largest number of restaurants?

```
pipeline = [
    {"$match":{"tags.amenity":"restaurant","address.city":{"$exists":1} }},

    {"$group":{"_id":"$address.city",
              "count":{"$sum":1}
           }},
    {
        "$sort":{
            "count":-1
           }
    }
]
projection = {"adress.city":1}
db.boston.aggregate(pipeline)
[doc for doc in db.boston.aggregate(pipeline)][:5]

[{u'_id': u'Boston', u'count': 54},
 {u'_id': u'Cambridge', u'count': 53},
 {u'_id': u'Somerville', u'count': 21},
 {u'_id': u'Brookline', u'count': 10},
 {u'_id': u'Allston', u'count': 5}]
```

It is not surprising that there are 54 and 53 restaurants in Boston and Cambridge, respectively. These two towns are the most famous universities and institutions are located, such as longwood medical area, MIT and Harvard.

Amenity-schools
Which town has the largest number of schools?

```
pipeline = [
    {"$match":{"tags.amenity":"school",
              "address.city":{"$exists":1}
              }
     },

    {"$group":{"_id":"$address.city",
              "count":{"$sum":1}
              }
     },
    {
        "$sort":{
            "count":-1
            }
    }
]
projection = {"adress.city":1}
db.boston.aggregate(pipeline)
[doc for doc in db.boston.aggregate(pipeline)][:5]

[{u'_id': u'Quincy', u'count': 8},
 {u'_id': u'Belmont', u'count': 6},
 {u'_id': u'Hyde Park', u'count': 5},
 {u'_id': u'Mattapan', u'count': 5},
 {u'_id': u'Roslindale', u'count': 4}]
```

It is Quincy, which is 10 miles south of Boston, and the number of schools for each town is kind of close. Different from hospitals and restaurant, school system are kind of evenly spread out, and the town with most school is not Boston.

3. Additional Ideas

For the address, the data are cleaned and then loaded into mongo db. Now that the data are in mongo db, using the Mongo db query, we can do similar auditing here. We can use query to see the phone number format. The further work we can do is to clean the phone number into a uniform format.

I wanted to know how many documents with phone number entries. I did the following query:

```
query = {"tags.phone":
        {"$exists":1}
        }
print db.boston.find(query).count()
```

Using this query I found 1066 records.

Here I use regular expression to find phone number that are not in the common format, the common format I defined is expressed in this regular expression:

```
regex = "^[0-9]{3}-[0-9]{3}-[0-9]{4}$"
```

How many are not in the “defined ” format expressed in the regular expression?

```
query = {"tags.phone":
        {"$exists":1,"$not":re.compile(regex)}
        }
db.boston.find(query).distinct( "tags.phone" )
```

Using this query I found 427 records, which means the regular expression I used above, there are about 40% of the phone number are not included.

Potential challenge:
Our purpose is to clean the phone number to let the phone number be uniform. The question here is which form to choose? Seems like there are different taste for choosing the phone format, it will be hard to reach a universally accepted uniform.