OpenStreetMap Sample Project
Data Wrangling with MongoDB

Ahmet Hamza EMRA

Map Area: Houston, TX, United States

https://www.openstreetmap.org/relation/2688911#map=10/29.7721/-95.3229

1. Problems Encountered in the Map

After initially downloading a small sample size of the Houston area and running it against a provisional data.py file, I noticed three main problems with the data, which I will discuss in the following order:

- Over-abbreviated street names or miss spellings corrected.
- “Incorrect” tag types corrected or removed
- “Incorrect” postal codes (*Houston area zip codes all begin with “77” however a large portion of all documented was not in good format.*)

Street Names

Before convert document to JSON format, I chance and correct miss spellings and other wrong things corrected with the python code. The code basically correct name for example “St. to Street. Original code can be found in other notes and quizzes folder (Named “Case Study_Reviewing street names.ipynb)
Code:

```
def update_name(name, mapping):
    for v in mapping:
        #print v
        if re.search(v,name):
            name=name.replace(v,mapping[v])
            return name
        else:
            return name

mapping = { "St": "Street",
            "St.": "Street",
            "Ave": "Avenue",
            "Rd.”: "Road",
           'Ave.': 'Avenue',
           'Blvd' : 'Boulevard',
           'Blvd.':'Boulevard',
           'Dr': 'Drive',
```

```
            'Frwy':'Freeway',
            'Pkwy': 'Parkway',
            'Rd':'Road',
            'Rd.':'Road',
            'Stree':'Street',
            'blvd':'Boulevard',
            'street':'Street'
             }
```

“Incorrect” tag types corrected or removed

There were a lot of “FIXME” tags, I checked them and most of them was questions or nonsense inputs from people. I remove them and then I convert XML document to JSON format document.

Incorrect postal codes:

There were 3 types of postal code input type. First one was plain 5-digit number the other ones was in order:

TX 77032

77032-232

So while converting the document I chance zip codes accordingly. For first type I check if the length of it is eight then I only take the last five digits. On the other hand, if length is 10 than I take first 5 digits.

Code:

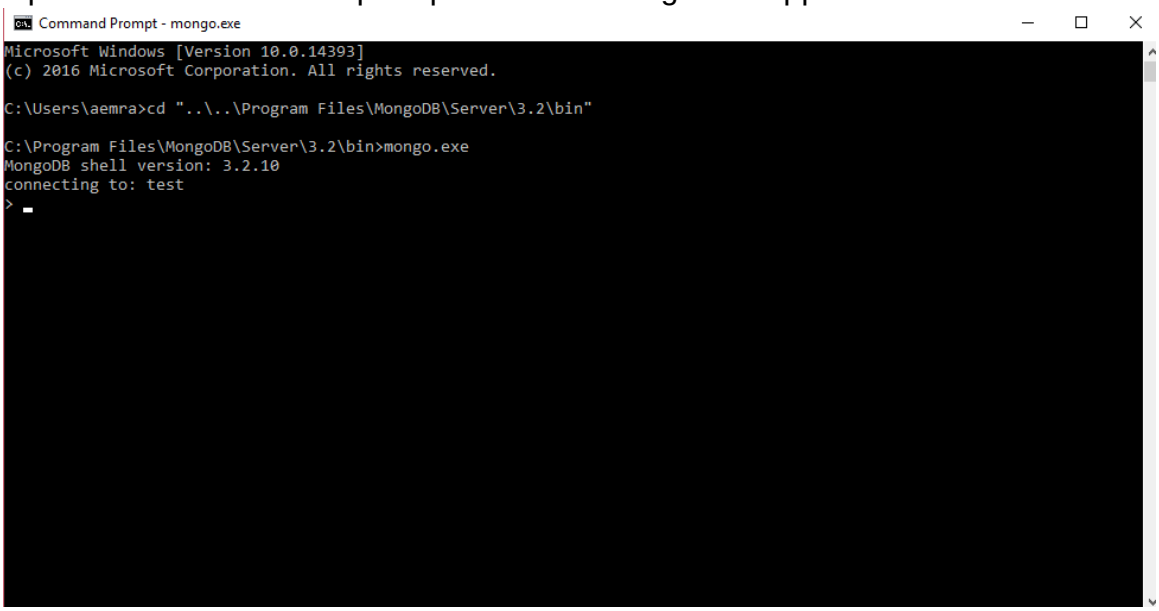
```
def update_zipcode(zipcode):
    zipcode=list(zipcode)
    if len(zipcode)==5:
        pass
    elif len(zipcode)==8:
        zipcode=zipcode[2:]
    elif len(zipcode)==10:
        zipcode=zipcode[:5]
    else:
        pass
    return int(''.join(zipcode))
```

2. Uploading to MongoDB

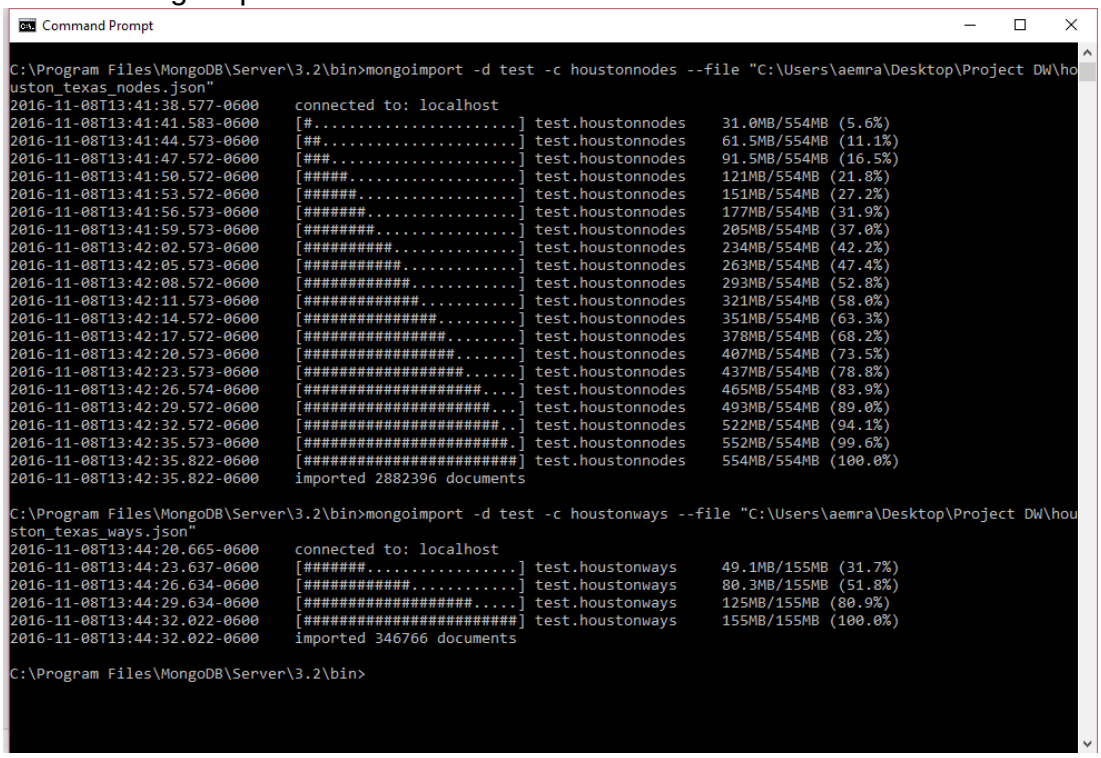
For me it was the most complicated part. I prefer to upload with mongo shell.

Steps:

1. Open cmd and start mongod.exe application
2. Open another command prompt and start mongo.exe application



3. Form another command prompt go to bin directory and use mongo import code
   Code: “mongoimport -d test -c new –file C:\..\..\..\foldername”



3. Data Overview

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

File sizes

```
Houston_texas.osm              643 MB
Houston_texas_ways.json        159 MB
Hoston_texas_nodes.json        567 MB
```

#Number of Documants

```
> db.houstonnodes.find().count()
2882396
> db.houstonways.find().count()
365469
```

#Number of unique users input (limited to 10):

Nodes:

```
In [39]: q6=[doc for doc in db.houstonnodes.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                                                       {"$sort":{"count":-1}}, {"$limit":10}])]
         pprint.pprint(q6)

[{'_id': 'woodpeck_fixbot', 'count': 588574},
 {'_id': 'TexasNHD', 'count': 543274},
 {'_id': 'afdreher', 'count': 376692},
 {'_id': 'scottyc', 'count': 172487},
 {'_id': 'cammace', 'count': 165707},
 {'_id': 'brianboru', 'count': 114565},
 {'_id': 'claysmalley', 'count': 93469},
 {'_id': 'RoadGeek_MD99', 'count': 69860},
 {'_id': 'skquinn', 'count': 64037},
 {'_id': 'xsintrik', 'count': 48721}]
```

Ways:

```
In [40]: q6=[doc for doc in db.houstonways.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                                                       {"$sort":{"count":-1}}, {"$limit":10}])]
         pprint.pprint(q6)

[{'_id': 'afdreher', 'count': 73034},
 {'_id': 'balrog-kun', 'count': 33825},
 {'_id': 'scottyc', 'count': 32250},
 {'_id': 'DaveHansenTiger', 'count': 29565},
 {'_id': 'cammace', 'count': 28135},
 {'_id': 'skquinn', 'count': 16956},
 {'_id': 'RoadGeek_MD99', 'count': 13553},
 {'_id': 'brianboru', 'count': 12761},
 {'_id': 'claysmalley', 'count': 12364},
 {'_id': 'Memoire', 'count': 7632}]
```

Number of users appearing only once (having 1 post)

```
In [42]: q10=[doc for doc in db.houstonnodes.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},
                                                        {"$group":{"_id":"$count", "num_users":{"$sum":1}}},
                                                        {"$sort":{"_id":1}}, {"$limit":1}])]
         pprint.pprint(q10)

[{'_id': 1, 'num_users': 237}]
```

Top 10 appearing amenities

```
In [45]: q11=[doc for doc in db.houstonnodes.aggregate([{"$match":{"amenity":{"$exists":1}}}, {"$group":{"_id":"$amenity",
         "count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":10}])]
         pprint.pprint(q11)
```

```
[{'_id': 'place_of_worship', 'count': 2211},
 {'_id': 'school', 'count': 1033},
 {'_id': 'fountain', 'count': 657},
 {'_id': 'restaurant', 'count': 636},
 {'_id': 'fast_food', 'count': 586},
 {'_id': 'fire_station', 'count': 355},
 {'_id': 'fuel', 'count': 253},
 {'_id': 'pharmacy', 'count': 171},
 {'_id': 'bank', 'count': 165},
 {'_id': 'police', 'count': 162}]
```

Biggest religion

```
In [11]: q7=[doc for doc in db.houstonnodes.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity":"place_of_worship"}},
                                                       {"$group":{"_id":"$religion", "count":{"$sum":1}}},
                                                      {"$sort":{"count":-1}}])]
         pprint.pprint(q7)
```

```
[{'_id': 'christian', 'count': 2150},
 {'_id': None, 'count': 23},
 {'_id': 'buddhist', 'count': 14},
 {'_id': 'jewish', 'count': 12},
 {'_id': 'muslim', 'count': 8},
 {'_id': 'unitarian_universalist', 'count': 4}]
```

Names of Muslim places

```
In [28]: q8=[doc for doc in db.houstonnodes.aggregate([{"$match":{"amenity":{"$exists":1}}},
                                                      {"$match":{"religion":"muslim"}},
                                                     {"$group":{"_id":"$name","count":{"$sum":1}}}])]
         pprint.pprint(q8)
```

```
[{'_id': 'Masjid As Sabireen', 'count': 1},
 {'_id': 'Makkah Masjid of Greater Houston', 'count': 1},
 {'_id': 'Isgh Islamic Society of Greater Houston', 'count': 1},
 {'_id': 'Islamic Society of Greater Houston', 'count': 5}]
```

Names of Christian places

```
In [46]: q8=[doc for doc in db.houstonnodes.aggregate([{"$match":{"amenity":{"$exists":1}}},
                                                      {"$match":{"religion":"christian"}},
                                                     {"$group":{"_id":"$name","count":{"$sum":1}}},
                                                     {"$limit":5}])]
         pprint.pprint(q8)
```

```
[{'_id': 'Pine Island Church', 'count': 1},
 {'_id': 'Advent Lutheran Church', 'count': 1},
 {'_id': 'Church of the Good Shepherd', 'count': 1},
 {'_id': 'Iglesia Bautista Houston', 'count': 1},
 {'_id': 'First Baptist Church of Patton Lakes', 'count': 1}]
```

Other queries can be found in Project_queries.ipynb

4. Additional Ideas

They put worship places which is very nice. On the other hand, dataset does not have any bus stop. They should include bus stop as well. They need to think about people who does not have car. Without any bus stop, coffeehouses, markets, nightclubs, etc. will meant nothing to those people, even they did pretty good job on adding these places.

Using one kind of input is good for analysis part but on the other side, for specific location I think 9 digits would be better. Because if one need to use United States Postal Service, then he or she need to put 9-digit. After my cleaning this kind of data will be lost. I think we need to keep +4 digit for this kind of benefits.

5. Conclusion

After this review of the data it's obvious that the Houston area is incomplete, though I believe it has been cleaned as much as possible but every time a person enters data, there will be errors. To get rid of this kind of errors, there should be input mask for some fields. For example, there should be only one type of zip code can be entered.