UDACITY_P3_FINAL

**********# Open StreetMap Final

DATA WRANGLING WITH MONGODB

by: Casey Iannone

Map Area: Chicago, Il, United States

MapZen Link to Extract

1 | Problems Encountered with Data

After creating a sample of the Chicago MapZen extract and auditing the street types I was able to find the over abbreviated streets (example: dr - Dr - dr.) as well as adjusting formatting to something that makes more sense to myself (example: s => South, n => North).

Here is a list of mappings I did through the sample file:

```
mapping = {"St": "Street", "St.": "Street", "street":
"Street","Dr": "Drive", "Dr.": "Drive","Rd": "Road","Ct":"Court"}
```

And here is a list of mappings that was appended to after several passes through the full dataset:

```
mapping = {
                    "St":
"Street","St.":"Street","street":"Street",
                    "st": "Street","Dr": "Drive", "Dr.":
"Drive","Ct":"Court","Blvd":"Boulevard","blvd":"Boulevard","Blvd.":
"Boulevard","Ct.":"Court","HWY":"Highway","Ln":"Lane","Ter":"Terrac
e","Ave.":"Avenue","Ave":"Avenue","road":"Road","rd":"Road","Rd.":"
```

```
Road","Rd":
"Road","place":"Place","US":"U.S.","Pl":"Place","Rte":"Route","IL":
"Illinois","W":"West","Trl":"Trail","avenue":"Avenue","Pkwy":"Parkw
ay","Cir":"Circle","N":"North","N.":"North","E":"East","S":"South",
"S":"South","S.":"South","W.":"West","E.":"East","W":"West"
          }
```

As you can see the auditing process for the street types continue to expand over each iteration.

2 | Data Overview

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

FILE SIZES:

- *Chicago.osm* 2.01 GB
- *Chicago.osm.json* 2.21 GB

MONGODB STATS:

```
{
u'storageSize': 727793664.0,
u'ok': 1.0,
u'avgObjSize': 248.1989511598606,
u'db': u'Final_osm',
u'indexes': 1,
u'objects': 9586971,
u'collections': 1,
u'numExtents': 0,
u'dataSize': 2379476147.0,
u'indexSize': 87064576.0
}
```

NUMBER OF DOCUMENTS

```
db.osm_final.find().count()
```

9586971

NUMBER OF NODES

```
db.osm_final.find({"type":"node"}).count()
```

8399395

NUMBER OF WAYS

```
db.osm_final.find({"type":"way"}).count()
```

1187370

TOP FIVE CONTRIBUTING USERS

```
def con_user():
    result = db.osm_final.aggregate([{"$group":
{"_id":"$created.user", "count":{"$sum":1}}},
                                     {"$sort":{"count":-1}},
                                     {"$limit":5}]
                                    )
    for a in result:
        pprint.pprint(a)


con_user()
```

1. {u‘_id’: u’chicago-buildings‘, u’count’: 5635534}
2. {u‘_id’: u’Umbugbene‘, u’count’: 1101042}
3. {u‘_id’: u’alexrudd (NHD)‘, u’count’: 232740}
4. {u‘_id’: u’woodpeck_fixbot‘, u’count’: 225702}
5. {u‘_id’: u’patester24‘, u’count’: 109091}

NUMBER OF USERS APPEARING ONLY ONCE (HAVING 1 POST)

```
def num_user():
    result = db.osm_final.aggregate([{"$group":
{"_id":"$created.user", "count":{"$sum":1}}},
                        {"$group":{"_id":"$count", "num_users":
{"$sum":1}}},
                        {"$sort":{"_id":1}}, {"$limit":1}])
    for a in result:
        pprint.pprint(a)

num_user()
```

{u'_id': 1, u'num_users': 503}

TOTAL NUMBER OF AMENITIES

(Amenities cover an assortment of community facilities including toilets, telephones, banks, pharmacies, schools and restaurants, etc....)

```
def amenities():
    result = db.osm_final.aggregate([{"$match":{"amenity":
{"$exists":True}}},
                        {"$group":{"_id":"$amenity", "count":
{"$sum":1}}},
                        {'$group':{'_id':'$amenity', 'count':
{'$sum':'$count'}}}
                         ])
    for a in result:
        pprint.pprint(a)

amenities()
```

{u'_id': None, u'count': 31821}

TOTAL NUMBER OF UNIQUE AMENITIES

```
def amenities_unique():
```

```
    result = db.osm_final.aggregate([{'$group':
{'_id':'$amenity','count':{'$sum':1}}},
                         {'$group':{'_id':'$amenity', 'count':
{"$sum":1}}}
                          ])
    for a in result:
        pprint.pprint(a)

amenities_unique()
```

{u'_id': None, u'count': 145}

TOP 10 AMENITY TYPES

```
def amenities_top():
    result = db.osm_final.aggregate([{'$match':{'amenity':
{'$exists':True}}},
                         {'$group':{'_id':'$amenity', 'count':
{"$sum":1}}},
                                     {'$sort':{'count':-1}},
                                     {'$limit':10}
                          ])
    for a in result:
        pprint.pprint(a)

amenities_top()
```

1. {u'_id': u'parking', u'count': 12577}
2. {u'_id': u'place_of_worship', u'count': 4305}
3. {u'_id': u'school', u'count': 3423}
4. {u'_id': u'restaurant', u'count': 1967}
5. {u'_id': u'fast_food', u'count': 1369}
6. {u'_id': u'fuel', u'count': 805}
7. {u'_id': u'bank', u'count': 595}
8. {u'_id': u'cafe', u'count': 455}
9. {u'_id': u'grave_yard', u'count': 450}
10. {u'_id': u'pharmacy', u'count': 370}

TOP CUISINE IN CHICAGO : BURGERS

```
def Cuisine_top():
    result = db.osm_final.aggregate([{'$match':{'cuisine':
{'$exists':True}}},
                        {'$group':{'_id':'$cuisine', 'count':
{"$sum":1},'name':{'$push':'$name'}}},
                                    {'$sort':{'count':-1}},
                                    {'$limit':1}
                         ])
    for a in result:
        pprint.pprint(a)

Cuisine_top()
```

***{u‘_id’: u’burger’,

u’count’: 494,

u’name’: [u“McDonald’s”,

```
       u'Char House Grill',
       u"Clarke’s”,} ***
```

3) Additional Ideas

TIMESTAMP:

Convert the time stamp from a string to a date time object when cleaning/reshaping the data. This would be done by adding the code here (pseudo-code):

```
for label in cursor.attrib:
    val = cursor.attrib[label]
    if label in CREATED:
```

```
        creadted[label]=val
        if label == 'lat' or label == 'lon':
        pos.append(float(val))
    if label == 'timestamp':
        convert label to datetime object
```

To implement the improvement you would first start by auditing that field to get an idea of how uniform it is and if there are any characters or other issues in making the conversion, much in the same way we did with the street type. You would then clean the field and write the converter, something I still would need to figure out how to do. The issues that could be experienced would be less than what was encountered with the street type audit, as this data is not user generated and that should cut down on non-uniform and incorrectly formatted data.

POSTAL CODES

An additional area that could use some auditing and cleaning would be the postal codes. After running the code below to examine the makeup of postal codes I noticed there are a few that are not properly formatted. For example:

Incorrect

- {u'_id': u'Wasco, IL 60183', u'count': 2}
- {u'_id': u'60189–8172', u'count': 2}

Correct

- {u'_id': u'60480', u'count': 2}
- {u'_id': u'60534', u'count': 2}
- {u'_id': u'60513', u'count': 2}

It does not appear to be a large number that are incorrectly formatted, but it is something that could help with additional aggregation and answering other questions. I would most use a similar method to what was used in the street type audit using regex to match a set number of characters and format.

```
def postal_codes():
```

```
    result  = db.osm_final.aggregate([{"$match":
{"address.postcode":{"$exists":1}}},
                  {"$group":{"_id":"$address.postcode", "count":
{"$sum":1}}},
                  {"$sort":{"count":1}}])
    for a in result:
        pprint.pprint(a)

postal_codes()
```

USER ENGAGEMENT

Given that OpenStreetMap data is open source and is contributed to by many individuals, it is important to examine who is creating and editing this data. While I think many would say that having few "cooks in the kitchen" would help with consistency, which I believe to be true, expanding the number of users contributing to the Chicago area could be helpful for accuracy of areas that a smaller group would not be able to cover.

In order to provide some context, lets take a look at the top 5 users and their contributions to see if this is something that is truly needed.

```
def users_5():
    result  = db.osm_final.aggregate([
            {'$match': {'created.user':{'$exists':1}}},
            {'$group': {'_id':'$created.user',
                        'count':{'$sum':1}}},
            {'$sort': {'count':-1}},
            {'$limit' : 5}
        ])
    for a in result:
        pprint.pprint(a)

users_5()
```

Total number of documents: *9,586,971*

Here is the percentage of documents contributed by the top five users:

- {u‘_id’: u’chicago-buildings‘, u’count’: 5635534} *59%*
- {u’_id‘: u’Umbugbene’, u’count’: 1101042} *12%*
- {u’_id‘: u’alexrudd (NHD)’, u’count’: 232740} *2.4%*
- {u’_id‘: u’woodpeck_fixbot’, u’count’: 225702} *2.4%*
- {u’_id‘: u’patester24’, u’count’: 109091} *1.1%*

You can see that 71% of the contributions come from just two users. Again, while this might be good for keeping things formatted correctly, I believe this takes away from OpenStreetMaps idea of local knowledge. I doubt those two users have the depth of knowledge needed to cover the vast landscape cover by the Chicago Illinois area. Therefore I believe promoting and refining the Go Map!! application can increase the number of users and frequency they contribute to the project. This could also be invaluable for cities or towns that are going through a development boon with changes occurring rapidly.