Udacity Data Analyst Nanodegree Text Wrangling Project

By: Bill Su

Date: Sept. 10th

Dataset Overview

This section is going to offer an overview of the dataset. First of all, the dataset is in XML format, and around 64.3 MB in size uncompressed in the XML format. In this section, I am going first load the dataset into a data frame and then extract information from the MongoDB server 1) number of unique users 2) number of nodes and ways 3) number of cafe and shops in the dataset.

This is a document recording data wrangling and cleaning process of openstreetmap dataset of Oxford, England (Where I studied abroad several years ago). This document is going to first present an audit the dataset as we have done in exercise 6 and clean the data. Then the dataset is going to be stored in Mongo DB. Finally, I am going to offer some insights regarding how the information obtained from this dataset could be used to support business decisions. Code below displays the initialized imports.

Data Auditing

K-Tag Examination

Just as what we have done in the exercises. I have first audited the "k" value for the tags in order to check whether they will be appropriate values for the resulting dictionary, without any problematic characters. The code to accomplish this is listed below, and the output displays two problematic headings I have found.

```
This is a problematic tag: fee:amount:box_van&minibus
This is a problematic tag: note 2
{'problemchars': 2, 'lower': 138656, 'other': 9028, 'lower_colon': 46833}
```

Problem for the first tag is that the sign & exists; the problem for the second one is that a blank space exists. I have decided that in order to solve this, I will simply remove problematic characters. Repaired tags are displayed below.

```
This is a repaired problem tag: fee:amount:box_vanminibus
This is a repaired problem tag: note2
```

Address Examination
I have written code to extract all tags and check whether all addresses are of the same form. Below displays the result of this process. This output only display name with greater than 1 occurrences to save spaces. I see two major problems for the end of addresses. The first problem is abbreviations. For example, Rd. needs to be Road, Ave needs to be avenue. The second major problem is punctuations. As much as I love “Way?”, or “Giles'”, it is not a good idea to have punctuations at the end of an address, it just confuses people. The third problem is that there are also some addresses that are not properly cased. For example, “road” need to be converted into “Road” in the database. For other types of address problems, I have decided to leave them be since they are not as significant as the three explained above.

```
['Lodge', '3SU', '3SP', '3SE', '3SA', '2LG', 'Cottages', '3RG', '1RG', 'Pavillion', '1RQ', '2', '8RZ', 'B
ank', 'Driftway', '3UB', 'Kidlington', 'Row', '4DH', 'Building', '54', '50', '52', '3TG', '3TA', '6AA', '0
BY', 'Ho', 'School', '1BU', '1BN', '1PA', 'Gardens', 'Crescent', '1', 'View', '1AB', 'Barn', '1AH', 'Rd'
, 'Witney', '2HS', '2HE', '3DY', '3DH', '12', '14', '17', '16', '18', 'Grange', '2ET', 'Hall', '1RX', '3', '60'
, '62', '64', '65', '67', '69', 'Wood', '4', '1UR', '1UH', '2DU', '2DP', '2DR', 'house', 'House', 'Close', '3
HB', '4AP', '4AQ', '4AG', '4AH', 'Acres', '6', 'Parade', '3PH', '2ER', '2EP', '2EW', '8LA', '24', '26', '
20', '28', '2a', '1EA', 'End', '1EJ', '7', '4BJ', '2BY', '2BS', '2BQ', '77', '73', '79', '2PB', 'Hill', '3JP', '8',
"Giles'", 'Park', 'Meadow', 'Way', '1DA', 'College', 'Church', '9', 'Terrace', 'Vicarage', 'Office', 'Acr
e', 'Furze', '3LW', '3LN', 'Roundabout', '3LJ', '3LD', '3LB', '38', '32', '36', '34', '3JS', '1XX', '1JH', '
1JJ', '1JP', 'Mews', '2AG', '6NA', 'Centre', 'Farm', '3QY', '3QX', '3QP', '4HT', '4HS', 'Cottage', '2N
A', '2NS', '3AZ', '3PW', '3PB', '3PD', '3PG', '3QG', 'Oxford', '1-8', '3ND', '5', '1HR', '1HU', '259', '4
8', '46', '44', '42']
```

Output after cleaning:

```
['Lodge', '3SU', '3SP', '3SE', '3SA', '2LG', 'Cottages', '3RG', '1RG', 'Pavillion', '1RQ', '2', '8RZ',
'Bank', 'Driftway', '3UB', 'Kidlington', 'Row', '4DH', 'Building', '54', '50', '52', '3TG', '3TA', '6AA',
'0BY', 'Ho', 'School', '1BU', '1BN', '1PA', 'Gardens', 'Crescent', '1', 'View', '1AB', 'Barn', '1AH',
'Witney', '2HS', '2HE', '3DY', '3DH', '12', '14', '17', '16', '18', 'Grange', '2ET', 'Hall', '1RX', '3', '60',
'62', '64', '65', '67', '69', 'Wood', '4', '1UR', '1UH', '2DU', '2DP', '2DR', 'House', 'Close', '3HB',
'4AP', '4AQ', '4AG', '4AH', 'Acres', '6', 'Parade', '3PH', '2ER', '2EP', '2EW', '8LA', '24', '26', '20',
'28', '2a', '1EA', 'End', '1EJ', '7', '4BJ', '2BY', '2BS', '2BQ', '77', '73', '79', '2PB', 'Hill', '3JP', '8',
'Park', 'Meadow', 'Way', '1DA', 'College', 'Church', '9', 'Terrace', 'Vicarage', 'Office', 'Acre',
'Furze', '3LW', '3LN', 'Roundabout', '3LJ', '3LD', '3LB', '38', '32', '36', '34', '3JS', '1XX', '1JH',
'1JJ', '1JP', 'Mews', '2AG', '6NA', 'Centre', 'Farm', '3QY', '3QX', '3QP', '4HT', '4HS', 'Cottage',
'2NA', '2NS', '3AZ', '3PW', '3PB', '3PD', '3PG', '3QG', 'Oxford', '1-8', '3ND', '5', '1HR', '1HU',
'259', '48', '46', '44', '42']
```

Notice that most lower cases are gone, I have decided to leave the letter and number endings alone because they are standard addresses.

Solving ZipCode Inconsistency

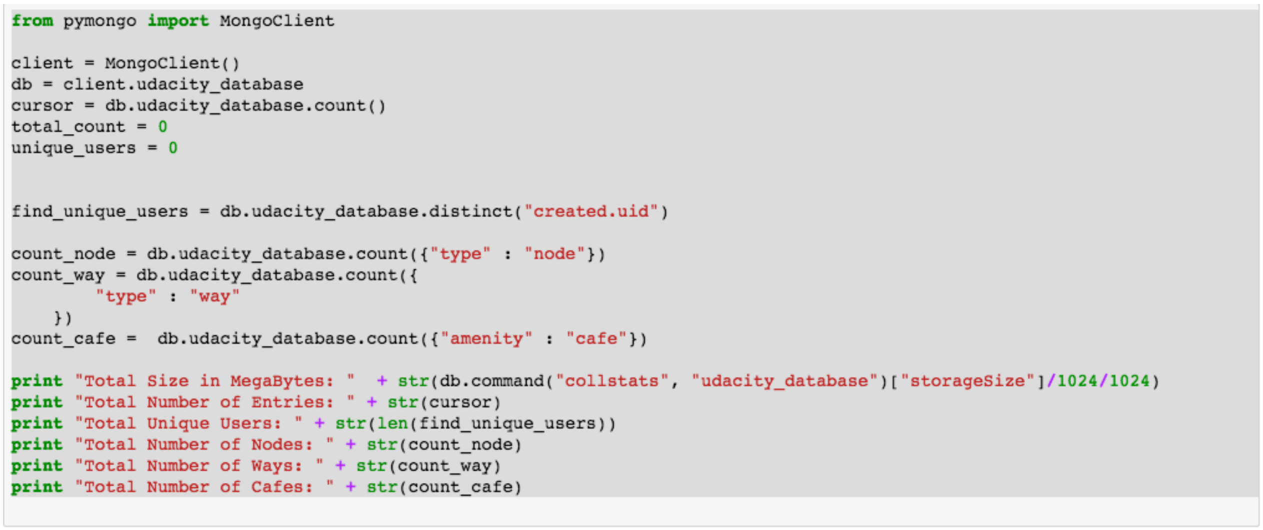
Another problem with the data at hand is that the zip codes are inconsistent. Below display a snippet of zipcode in the openstreetmap dataset.

OX2
OX2
OX13 6RB
OX5 1DZ
OX29
OX2
OX2
OX3
OX2

In UK, postal codes comes in the format in which the first three or four digits represent the area and district of the location. For example, OX2 would mean Oxfordshire district 2. Sometimes postal code also comes with three more letters, representing the sector and unit of the location, which is more detailed. Because some addresses provided in OSM do not have last three figures, I have decided to unify all postal codes to district only. Post-removal of sector and unit postal code, a snippet of the postal code is presented below.

OX2
OX2
OX13
OX5
OX29
OX2
OX2

Storage and Data Overview in MongoDB
After cleaning, the entire file is converted into dictionary, and then added to MongoDB. Then we have performed several operations to grant an overview of the dataset. All of the overviews are displayed below; I have also provided queries for how those results are generated.



```python
from pymongo import MongoClient

client = MongoClient()
db = client.udacity_database
cursor = db.udacity_database.count()
total_count = 0
unique_users = 0


find_unique_users = db.udacity_database.distinct("created.uid")

count_node = db.udacity_database.count({"type" : "node"})
count_way = db.udacity_database.count({
        "type" : "way"
    })
count_cafe =  db.udacity_database.count({"amenity" : "cafe"})

print "Total Size in MegaBytes: "  + str(db.command("collstats", "udacity_database")["storageSize"]/1024/1024)
print "Total Number of Entries: " + str(cursor)
print "Total Unique Users: " + str(len(find_unique_users))
print "Total Number of Nodes: " + str(count_node)
print "Total Number of Ways: " + str(count_way)
print "Total Number of Cafes: " + str(count_cafe)
```

Total Size in MegaBytes: 166
Total Number of Entries: 900509
Total Unique Users: 514 Total
Number of Nodes: 267034 Total
Number of Ways: 45111
Total Number of Cafes: 104

More Cleaning Ideas and Implementation Suggestions

In this document, due to restrain of space and time, we only made sure addresses and dictionary tags are in the correct format. The goal of this section is to briefly talk about what else can be done in cleaning the data, and how this data could be implemented and analyzed for realistic results.

Removal of Non-Attributive Locations for Shorter Dataset

If we look at the dataset, we will realize that there are two types of data: the location data in which no information are attached to it, and the information data which references to the location nodes. Even though this is important when storing in large scale, I believe the data would be cleaner if those location nodes are merged with information nodes. The implementation below does that.

Potential Implementation Ideas

A dataset is considered useless if it has no real-world implementation value. As the reader may know, Oxford is home to oldest and probably one of the largest university of the world "Oxford University". According my personal experience traveling in Oxford, the university is a pain to navigate around with because it is consist of 38 constituent collages. The idea is to geo-map all oxford universities and provide users with a sample mapping application of where all the universities are (total not copying Google map). A simple implementation example are displayed below, which locates all university related structured in Oxford.

```
[[51.7515791, -1.2508002], [51.7515338, -1.2510764], [51.7515217, -1.2511969], [51.7515672,
-1.2512094], [51.7516116, -1.2512183], [51.7516798, -1.2512289], [51.7518827, -1.2512708], [
51.75186, -1.2513385], [51.7518097, -1.2517588], [51.7517966, -1.2518665], [51.7517589, -1.2
518626], [51.751731, -1.2521152], [51.7517163, -1.2522242], [51.7517569, -1.2522424], [51.75
17685, -1.2522509], [51.751791, -1.2522693], [51.7518323, -1.2522898], [51.7518009, -1.2525
13], [51.7517523, -1.2525261], [51.751727, -1.2527262], [51.7516774, -1.2529917], [51.751766
9, -1.2530323], [51.7518302, -1.2530456], [51.7518466, -1.2527372], [51.752026, -1.2527745],
[51.7520311, -1.2527391], [51.752097, -1.2527491], [51.7522142, -1.2527671], [51.7522874, -1
.2527782], [51.7522724, -1.2528268], [51.7523094, -1.2528504], [51.7523057, -1.2528787], [51
.752387, -1.2529039], [51.7523815, -1.2529449], [51.7525031, -1.2529669], [51.7525, -1.25302
3], [51.7525507, -1.2530376], [51.7525682, -1.2528671], [51.7525822, -1.2527475], [51.752611
2, -1.2525313], [51.7526253, -1.2523349], [51.7526283, -1.2522212], [51.752631, -1.2521199],
[51.7526344, -1.251999], [51.752635, -1.2519801], [51.7526384, -1.2518594], [51.7526418, -1.
2517608], [51.7526486, -1.2517506], [51.7526486, -1.2517075], [51.7526415, -1.2517002], [51.
7526396, -1.2516639], [51.75263, -1.2515047], [51.7526278, -1.2514755], [51.7526148, -1.251
3005], [51.752603, -1.2511419], [51.7525913, -1.2510843], [51.7525356, -1.2507232], [51.7524
853, -1.2503977], [51.7523555, -1.2504652], [51.752279, -1.250505], [51.7522266, -1.2505321]
, [51.7521816, -1.2505479], [51.7520728, -1.2505862], [51.7520713, -1.2505801], [51.7520636,
 -1.2505485], [51.7520135, -1.250572], [51.7520152, -1.2505833], [51.7520205, -1.2506268], [5
1.752026, -1.2506789], [51.7520299, -1.25072], [51.7519534, -1.2507455], [51.7519167, -1.250
7591], [51.7519154, -1.2507593], [51.7518969, -1.2507609], [51.75188, -1.2506517], [51.75184
08, -1.2506654], [51.7518263, -1.250553], [51.7518589, -1.2505432], [51.7517489, -1.2497229]
, [51.7517455, -1.2497234], [51.7517184, -1.2497273], [51.751681, -1.2497373], [51.751633, -1
.2497457], [51.7516238, -1.2498821], [51.7516182, -1.2499979], [51.7516122, -1.2501193], [51
.7516073, -1.2502208], [51.7516065, -1.2502372], [51.7516024, -1.2503215], [51.7516014, -1.2
50341], [51.7515955, -1.2504623], [51.7515791, -1.2508002]]
```

Above display all positions related to University College. The data above have actually presented a challenge for us. There are too many locations associated with University College, therefore it would be difficult to decide which point/area on the map is actually University College. One implementation solution to this problem is to try to find the four corners of the location and form a square area that represents the college. However, the square area might not accurately represent the actual shape of the university, and might mislead students when they are trying to find the place.

Finally, another possible solution is to use a geo-mapping package that would map out each location's boarder according to all coordinates, but the computational cost will be great in implementing such solution, and also take a lot more effort than the square solution presented above.

Conclusion

In this project we have first audited the data and corrected tags and addresses for the dictionary. Then I have entered data into MongoDB and did some brief overviews of the dataset. Finally, I have made some implementation recommendations and additional cleanings to make the data more concise and easier to implement.