Data Wrangling with MongoDB

by Joy Lal Chattaraj

Step 1 : The Data Set

- Map Area : Mumbai, India
- Source : http://www.openstreetmap.org/
- File Sizes : Compressed : 6.5MB , Uncompressed : 92MB

Importing the dataset location

```
import xml.etree.cElementTree as ET
from pprint import pprint

mumbai_file = "../Data sets/mumbai.osm"
```

Step 2 : Checking for problems and inconsistency in the data

Finding the types of Parent Tags

```
{'bounds': 1, 'node': 470060, 'relation': 126, 'way': 40396}
```

Here we see that there a broadly four types of tags: nodes, way, relation and bounds Since number of bounds and relations are far lesser, we need to explore nodes and ways thoroughly

```
node defaultdict(<type 'set'>, {'tag': set(['k', 'v'])})
relation defaultdict(<type 'set'>, {'member': set(['role', 'ref', 'type']), 'tag': set(['k', 'v'])})
bounds defaultdict(<type 'set'>, {})
way defaultdict(<type 'set'>, {'tag': set(['k', 'v']), 'nd': set(['ref'])})
```

Further, we look upon the various child tags present in the root tags we saw in the previous code snippet's result. The only thing left now is to check for characters that can cause problem and modify those names to shape our data.

Fields with sub-fields and problematic characters

Problematic Characters

```
Cable TV Provider
EState Consultants
Sahakari Bhandar
business Park
business Park
Mangeshi Dham
street parking
street parking
street parking
street parking
street parking
street parking
street parking
Community Centre
Area details
street parking
street parking
Golden Park
maneshi dham
mangeshi dham
{'lower': 108481, 'lower_colon': 3270, 'other': 842, 'problemchars': 20}
```

There isn't any problematic character except the 'space' character, which is not much of a concern. The only point of interest is the ':' character which means the presence of sub fields.

A final check on the field names to view all the field names and make decisons for cleaning accordingly

```
set(['AND:importance_level',
     'AND_a_c',
     'AND_a_i',
     'AND_a_nosr_p',
     'AND_a_nosr_r',
     'AND_a_w',
     'AREA',
     'Amenity',
     'Area details ',
     'Cable TV Provider',
     -
     -
     ---------OUTPUT HIDDEN---------
```

As we notice, there are a few abbreviated field names and some have been repeated like addr:postcode and postcode, which need to be taken care of.

Scanning Amenities for checking consistency in field type

```
set(['Canteen',
     'Educational Complex',
     'Gym',
     'Gymkhana',
     'Society',
     'Workshop
     -
     -
     ---------OUTPUT HIDDEN---------
```

We notice that there are a few entries where there are multiple input values in the form of ';' seperated values.

Mapping correct names for fields having sub fields.

List of all fields having sub fields

```
{'AND': 'AND',
 ---------OUTPUT HIDDEN---------
```

all over-abbreviated words shall be converted to their full names

```
{'AND': 'AND',
 'addr': 'address',
 'building': 'building',
 'fuel': 'fuel',
 'gns': 'GEOnet Name Server',
 'internet_access': 'internet_access',
 'is_in': 'located_in',
 'name': 'name',
 'oneway': 'oneway',
 'payment': 'payment',
 'place': 'place',
 'ref': 'reference',
 'seamark': 'seamark',
 'ship': 'ship',
 'shop': 'shop',
 'source': 'source',
 'tower': 'tower',
 'turn': 'turn',
 'wp': 'wp'}
```

The 'keys_with_sub_types' variable contains the correct name for each of the fields which contain multiple sub fields.

Understanding the name field

```
['',
 'bn',
 'cs',
 ---------OUTPUT HIDDEN---------
```

The name field also has sub fields which include names in various other languages, the language codes can be converted to their respective language names.

```
{'': 'other',
 'bn': 'bengali',
 'cs': 'czech',
 'de': 'german',
 'en': 'english',
 'es': 'spanish',
 'fr': 'french',
 'gu': 'gujrati',
 'hi': 'hindi',
 'jbo': 'lobjan',
 'kn': 'kanada',
 'ma': 'arabic',
 'mr': 'marathi',
 'pl': 'polish',
 'pt': 'portuguese',
 'ru': 'russian',
 'sk': 'slovak',
 'sr': 'serbian',
 'ta': 'tamil',
 'te': 'telugu'}
```

Language mapping variable contains the name of language for its short code.

Postcodes

```
set(['400001',
     -
     -
     ---------OUTPUT HIDDEN---------',
     '400 022',
     '400 601',
    -
     -
     ---------OUTPUT HIDDEN---------
     '400076',
     '400076, India',
     '400077',
     -
     -
     ---------OUTPUT HIDDEN---------
```

Problems in Postcode variables 1. codes seperated by space in some cases 2. codes contain alphabets in some cases 3. multiple keys where postal codes can be found : addr:postcode, postal_code and postcode

Step 3: Shaping data into Python Dictionary

Desired structure of the dictionary:

```
{
    id : ,
    type : ,
    visible : ,
    created : {
                version : ,
                changeset : ,
                timestamp : ,
                user : ,
                uid :
              },
    pos : [latitude, longitude],
    address : {
                housenumber: ,
                postcode : ,
                street :
               },
    amenity : ,
    cuisine : ,
    name : ,
    phone : ,
    .
    .
    _other fields_
    .
    .
  }
```

Now we can start putting the data from xml to a python dictionary and then make a list of dictionaries that can be stored into a json file. While creating the dictionary, we shall take care of the problems we detected earlier.

Save the list of dictionaries to a json file

```
import json
with open('data.json', 'w') as file:
    json.dump(final_dictionary, file)
```

Comparing sizes of the file before and after

```
Size of Json File:  105 MB
Size of uncompressed osm File :  92 MB
```

Step 4: Inserting into MongoDB and data overview

```
import pymongo

connection = pymongo.MongoClient("mongodb://localhost")

db = connection.osm_data

record = db.mumbai_data

mumbai_data = open('data.json', 'r')

parsed_mumbai_data = json.loads(mumbai_data.read())

for entry in parsed_mumbai_data:
    record.insert_one(entry)
```

Overview of the data

Objects in MongoDB

```
510456
```

Dictionaries in the list

```
510456
```

Number of Nodes and Ways

```
[{u'_id': u'node', u'count': 470053}, {u'_id': u'way', u'count': 40213}]
```

Number of distinct users

```
455
```

Step 5: Further exploration of data using MongoDB

### Top 20 Contributers

```
[{u'_id': u'PlaneMad', u'count': 65571},
 {u'_id': u'MJL Wood', u'count': 61257},
 {u'_id': u'balaji88', u'count': 59941},
 {u'_id': u'parambyte', u'count': 45287},
 {u'_id': u'udaya', u'count': 44663},
 {u'_id': u'smith_dsm', u'count': 39375},
 {u'_id': u'Giyavudeen', u'count': 30007},
 {u'_id': u'indigomc', u'count': 19682},
 {u'_id': u'shekhar', u'count': 13296},
 {u'_id': u'Moorthy1', u'count': 11864},
 {u'_id': u'singleton', u'count': 10324},
 {u'_id': u'PremK', u'count': 10180},
 {u'_id': u'dmgroom_coastlines', u'count': 7880},
 {u'_id': u'gaurav jain', u'count': 7754},
 {u'_id': u'Heinz_V', u'count': 6907},
 {u'_id': u'Oberaffe', u'count': 6393},
 {u'_id': u'Meghanand', u'count': 6135},
 {u'_id': u'Shekhar11', u'count': 4717},
 {u'_id': u'jain zachariah', u'count': 4545},
 {u'_id': u'katpatuka', u'count': 4402}]
```

We can see a highly skewed distribution in terms of contribution

List of Cuisines in Mumbai

```
[{u'_id': u'italian', u'count': 5},
 {u'_id': u'indian', u'count': 30},
 {u'_id': u'coffee_shop', u'count': 26},
 {u'_id': u'mediterranean', u'count': 1},
 {u'_id': u'burger', u'count': 12},
 {u'_id': u'pizza', u'count': 10},
 {u'_id': u'persian', u'count': 1},
 {u'_id': u'vegetarian', u'count': 14},
 {u'_id': u'seafood', u'count': 2},
 {u'_id': u'sandwich', u'count': 4},
 {u'_id': u'regional', u'count': 3},
 {u'_id': u'chinese', u'count': 7},
 {u'_id': u'chicken', u'count': 2},
 {u'_id': u'american', u'count': 2},
 {u'_id': u'ice_cream', u'count': 2},
 {u'_id': u'asian', u'count': 1},
 {u'_id': u'international', u'count': 5},
 {u'_id': u'thai', u'count': 1},
 {u'_id': u'spanish', u'count': 1},
 {u'_id': u'sad_food', u'count': 1},
 {u'_id': u'Goan', u'count': 1}]
```

The numbers although aren't enough to draw conclusions, but it's still not a matter of surprise that Indian cuisine is the highest in number.

Top 20 Amenities in Mumbai

```
[{u'_id': [u'place_of_worship'], u'count': 246},
 {u'_id': [u'school'], u'count': 211},
 {u'_id': [u'restaurant'], u'count': 161},
 {u'_id': [u'bank'], u'count': 134},
 {u'_id': [u'parking'], u'count': 117},
 {u'_id': [u'hospital'], u'count': 110},
 {u'_id': [u'bus_station'], u'count': 102},
 {u'_id': [u'fuel'], u'count': 100},
 {u'_id': [u'college'], u'count': 83},
 {u'_id': [u'fast_food'], u'count': 66},
 {u'_id': [u'police'], u'count': 57},
 {u'_id': [u'cafe'], u'count': 55},
 {u'_id': [u'atm'], u'count': 51},
 {u'_id': [u'cinema'], u'count': 50},
 {u'_id': [u'swimming_pool'], u'count': 48},
 {u'_id': [u'post_office'], u'count': 40},
 {u'_id': [u'pharmacy'], u'count': 36},
 {u'_id': [u'toilets'], u'count': 35},
 {u'_id': [u'marketplace'], u'count': 27},
 {u'_id': [u'public_building'], u'count': 22}]
```

Its quite surprising that there are so many places of worship!, other figures are as expected.

Top 10 types of buildings in Mumbai

```
[{u'_id': u'yes', u'count': 6222},
 {u'_id': u'residential', u'count': 485},
 {u'_id': u'apartments', u'count': 305},
 {u'_id': u'office', u'count': 76},
 {u'_id': u'train_station', u'count': 58},
 {u'_id': u'house', u'count': 57},
 {u'_id': u'industrial', u'count': 53},
 {u'_id': u'concourse', u'count': 40},
 {u'_id': u'commercial', u'count': 17},
 {u'_id': u'roof', u'count': 9}]
```

Lots of Residential buildings and appartments, nothing surprising as Mumbai is one of India's most populated cities.

Top 10 Postcodes in the area

```
[{u'_id': u'400057', u'count': 135},
 {u'_id': u'400053', u'count': 63},
 {u'_id': u'400050', u'count': 56},
 {u'_id': u'400061', u'count': 30},
 {u'_id': u'400607', u'count': 29},
 {u'_id': u'400049', u'count': 12},
 {u'_id': u'400071', u'count': 8},
 {u'_id': u'400088', u'count': 8},
 {u'_id': u'400601', u'count': 7},
 {u'_id': u'400058', u'count': 7}]
```

Conclusion and other ideas about the dataset

When we audit the data, it is quite clear that although there are minor errors caused by human input, the dataset is fairly well-cleaned. Considering there are lots of contributors for this map, there are a great number of human errors in this project. I'd recommend a structured input form so everyone can input the same data format to reduce this error or we can create a more robust script to clean the data on a regular basis. The structured input will make the cleaning of data much easier. We can even specify data types and constraints on certain fields thus making comparison easier and maintaining the integrity of the data. The above said solution is a bit difficult to implement. It would require a person to dedicate all of this time in writing scripts and designig the input struture, also users may find it difficult to submit data programatically because of such constraints. It might be a good idea to reward user's with more contributions in such senarios.