OpenStreetMap Project: Data Wrangling with MongoDB

Project Summary:

OpenStreetMap - http://www.openstreetmap.org, an open project to create a free map around the world, is a powerful tool for viewing maps. The initial data of the map were collected using a handheld GPS and a notebook, digital camera, or a voice recorder. These data heavily rely on the human input, which may cause inconsistent input, misspellings or error (e.g. inconsistent input of Street, street, st., Str).

This project is going to focus on map of Las Vegas - Nevada, USA and Wrangling the data.

- Overview data (code: mapparser.py)
- Check the "k" value for each "<tag>" (code filename: tags.py)
- Find out number of unique users contributed to the map (code filename: users.py)
- Fix unexpected street types (e.g. street, st., St. to be Street) (code filename: audit.py);
- Transform the shape of data and insert data into mongodb (code filename: data.py and mongodb.py)
- Use MongoDB queries to find number and names of hotels, and shopping malls (code filename: query.py).

The python codes mentioned above are included in a separated folder/files and tested by sample dataset (as a part of the whole dataset) before run for the whole dataset.

As with any user generated content, there is likely going to be dirty data. In this project I'll attempt to do some auditing, cleaning, and data summarizing tasks with Python and MongoDB.

Chosen Map Area (Data source)

For this project, I chose to analyze data from Las Vegas. It is one of the most popular tourist destinations in world and I would like to visit once in my lifetime. Also I would like to know this place well using this project. I hope this map to be improved in future.

I downloaded the data using following steps:

- Open URL - http://www.openstreetmap.org/
- Search "Las vegas"
- Click on "Export" button
- Selected "Metro Extract" - linked to "https://mapzen.com/metro-extracts/
- Las Vegas, Nevada (OSM XML) download
- Decompress the file to las-vegas_nevada.osm (169 MB)

Auditing the Data

1. With the OSM XML file downloaded, I have parse through it with ElementTree and find the number of each type of element. In this project, I had use iterative parsing as the XML download can be too large to work with in memory.

2. Here I have built three regular expressions: *lower*, *lower_colon*, and *problemchars*.
    - *lower:* matches strings containing lower case characters
    - *lower_colon:* matches strings containing lower case characters and a single colon within the string
    - *problemchars:* matches characters that cannot be used within keys in MongoDB
3. Within the node element there are two tag children. The key for both of these children begins with *addr:*. Later in this notebook I will use the lower_colonregex to help find these keys so I can build a single address document within a larger json document.
4. Now I will redefine process_map to build a set of unique userid's found within the XML. I will then output the length of this set, representing the number of unique users making edits in the chosen map area.

Problems encountered in the map

Street Names

The majority of this project devoted to auditing and cleaning street names seen within the OSM XML. Street types used by users in the process of mapping are quite often abbreviated. I have attempted to find these abbreviations and replace them with their full text form. **The plan of action is as follows:**

- Build a regex to match the last token in a string (with an optional '.') as this is typically where you would find the street type in an address
- Build a list of expected street types that do not need to be cleaned
- Parse through the XML looking for tag elements with k="addr:street" attributes
- Perform a search using the regex on the value of the v attribute of these elements (the street name string)
- Build a dictionary with keys that are matches to the regex (street types) and a set of street names where the particular key was found as the value. This will allow us to determine what needs to be cleaned.
- Build a second dictionary that contains a map from an offending street type to a clean street type
- Build a second regex that will match these offending street types anywhere in a string
- Build a function that will return a clean string using the mapping dictionary and this second regex

The first step is to build a regex to match the last token in a string optionally ending with a period. I have also built a list of street types I expect to see in a clean street name.

1. Over abbreviated Street Names: In this dataset, there are 7 types of problems in Street types. Some of them are inconsistent input, for example, "AVE", "Ave" and "Ave.". Some are abbreviation: "Ln.", "Pkwy", "St" and "Rd". Using attached auditing code (code filename-audit.py), these inconsistent inputs or abbreviations were fixed (table 1).

Table 1 Fixing problem of street types

Problem#	Before Auditing	After Auditing
1	AVE , Ave, Ave.	Avenue
2	Blvd, Blvd.	Boulevard
3	Dr Dr.	Drive
4	Ln.	Lane
5	Pkwy	Parkway
6	St	Street
7	Rd	Road

2. The keys were also validated (using code: tags.py)
In this dataset, there are 240883 keys including 'lower' regular expression ('^([a-z]|_)*$'), 263369 keys including 'lower_colon' regular expression ('^([a-z]|_)*:([a-z]|_)*$') and 2 including 'problemchars' expression ('[=\+/&<>;\'"\?%#$@\,\. \t\r\n]').
The keys with 'problemchars' expression will be ignored during data shape transformation (code: data.py).

Preparing for MongoDB

To load the XML data into MongoDB, I have to transform the data into json documents structured like this:

```
{
   "id": "2406124091",
   "type: "node",
   "visible":"true",
   "created": {
             "version":"2",
             "changeset":"17206049",
             "timestamp":"2013-08-03T16:43:42Z",
             "user":"linuxUser16",
             "uid":"1219059"
           },
   "pos": [41.9757030, -87.6921867],
   "address": {
             "housenumber": "5157",
             "postcode": "60625",
             "street": "North Lincoln Ave"
           },
   "amenity": "restaurant",
   "cuisine": "mexican",
   "name": "La Cabana De Don Luis",
   "phone": "1 (773)-271-5176"
}
```

The transform will follow these rules:

- Process only 2 types of top level tags: node and way
- All attributes of node and way should be turned into regular key/value pairs, except:
    - The following attributes should be added under a key created: version, changeset, timestamp, user, uid
    - Attributes for latitude and longitude should be added to a pos array, for use in geospacial indexing. Make sure the values inside pos array are floats and not strings.
- If second level tag "k" value contains problematic characters, it should be ignored
- If second level tag "k" value starts with "addr:", it should be added to a dictionary address
- If second level tag "k" value does not start with "addr:", but contains ":", you can process it same as any other tag.
- If there is a second ":" that separates the type/direction of a street, the tag should be ignored, for example:

```
<tag k="addr:housenumber" v="5158"/>
<tag k="addr:street" v="North Lincoln Avenue"/>
<tag k="addr:street:name" v="Lincoln"/>
<tag k="addr:street:prefix" v="North"/>
<tag k="addr:street:type" v="Avenue"/>
<tag k="amenity" v="pharmacy"/>

should be turned into:
        {
          "address": {
                    "housenumber": 5158,
                    "street": "North Lincoln Avenue"
                  },
          "amenity": "pharmacy"
        }

For "way" specifically:
<nd ref="305896090"/>
<nd ref="1719825889"/>

should be turned into:
        {
          "node_refs": ["305896090", "1719825889"]
        }
```

To do this transformation, I will define a function *shape_element* that processes an element. Within this function I will use the update function with the regexes and mapping dictionaries defined above to clean street addresses.

Overview of the data

To overview the data, the following items were recorded:

- Size of the file (las-vegas_nevada.osm): 169 MB
- Size of the json file (las-vegas_nevada.osm.json): 186 MB
- Number of unique users (code: users.py): contributed by 618 unique users
- Number of nodes and ways (code: mappraser.py): nodes: 723370, ways:78394;
- Number of top 10 amenities with most amount of units
  (attached code: query.py, function: get_query (data, db)):

[('parking', 746), ('school', 535), ('place_of_worship', 367), ('fountain', 269), ('restaurant', 200), ('fast_food', 158), ('fuel', 139), ('hotel', 104), ('fire_station', 70), ('hospital', 66)]

Additional data exploration using MongoDB queries:

1. Interesting point – found Hotels and malls in the datasets
I was looking for all the hotels and shopping malls in Las Vegas. To do that, I try to find hotels and malls listed as amenity using two pipelines (pipeline1 and pipeline3 in attached code: query.py, function: get_pipeline( )) to query database inserted into MongoDB. However, only one hotel without name and two malls were found, which does not make sense. I went back to check the original dataset and noticed that "hotel" and "mall" were listed as **"tourism" and "shop",** respectively.

Thus, I edited the query pipelines (pipeline2 and pipeline4) and got the following results: Hotels as amenity: 1, Hotels as tourism: 103; Malls as amenity: 2, Malls as shop: 8.

Finally, in order to improve the map, I modified the data.py code and added hotels and malls to amenity. Execute the query pipeline1 and pipeline3 again, and then I got 10 malls and 104 hotels as amenity. These results suggest the hotels and malls were successfully and correctly added into new database.

Pipelines:

```
#hotels as amenity
pipeline1 = [{"$match": {"amenity": "hotel"}},
{"$project":{"_id": "$name", "cuisine": "$cuisine", "phone": "$phone"}}]
#hotels as tourism
pipeline2 = [{"$match": {"tourism": "hotel"}},
{"$project":{"_id": "$name", "address": "$address", "phone": "$phone"}}]
#mall as amenity
pipeline3 = [{"$match": {"amenity": "mall"}},
{"$project":{"_id": "$name", "cuisine": "$cuisine", "phone": "$phone"}}]
#mall as shop
pipeline4 = [{"$match": {"shop": "mall"}},
{"$project":{"_id": "$name", "address": "$address", "phone": "$phone"}}]
```

Outputs:

Hotels as amenity: 104

{u'ok': 1.0,
u'result': [{u'_id': u'SLS Hotel and Casino', u'phone': u'(702) 737-2111'},
{u'_id': u'Four Seasons'},
{u'_id': u'Tuscany'},
{u'_id': u'Embassy Suites'},
{u'_id': u'Silver Sevens Hotel & Casino',
u'phone': u'(702) 733-7000'},
{u'_id': u'Suncoast Hotel and Casino'},
{u'_id': u'Lake Mead Lodge'},
{u'_id': u'Encore Casino at Wynn'},
{},
{u'_id': u'Hacienda Hotel and Casino',
u'phone': u'(702) 293-5000'},
{u'_id': u'Hilton Grand Vacations Suites on the Las Vegas Strip',
u'phone': u'702-765-8300'},
{u'_id': u'Hilton Grand Vacations Suites - Las Vegas (Convention Center)',
u'phone': u'702-946-9210'},
{u'_id': u'Embassy Suites Convention Center Las Vegas',
u'phone': u'702-893-8000'},
{u'_id': u'Renaissance Las Vegas Hotel',
u'phone': u'702-784-5700'},
{u'_id': u'Extended Stay America'},
{u'_id': u'Boulder Station Hotel and Casino'},
{u'_id': u'Element Las Vegas Summerline, 10555 Discovery Drive'},
{u'_id': u'Main Street Station Casino and Hotel'},
{},
{u'_id': u'Americas Best Value Inn'},
{u'_id': u'Summer Bay Resort'},
{u'_id': u'Candlewood Suites Las Vegas'},
{u'_id': u'Quality Inn'},
{u'_id': u'Boulder Dam Hotel'},
{u'_id': u'Boulder Inn and Suites'},
{u'_id': u'Circus Circus, 2880 Las Vegas Boulevard, NV 89109 Las Vegas, Vereinigte Staaten von Amerika'},
{u'_id': u'Fortune Hotel & Suites', u'phone': u'702-830-4010'},
{u'_id': u'Super 8'},
{u'_id': u'Stratosphere Hotel and Casino'},
{u'_id': u'Residence Inn Las Vegas Convention Center'},
{u'_id': u'Courtyard Las Vegas Convention Center'},
{u'_id': u'Hilton Garden Inn Henderson'},
{u'_id': u'Palms Casino Resort'},
{u'_id': u'Gold Coast Hotel & Casino'},
{},
{u'_id': u'Tropicana Hotel and Casino'},
{u'_id': u'Luxor Hotel and Casino'},
{u'_id': u'Mandalay Bay Casino & Resort'},
{u'_id': u'Polo Towers'},

{u'_id': u'Encore'},
{u'_id': u'Mirage Hotel and Casino'},
{u'_id': u'Treasure Island Hotel and Casino'},
{u'_id': u'Best Western Mardi Gras Inn'},
{u'_id': u'Plaza Hotel', u'phone': u'(702) 386-2110'},
{u'_id': u'Paris Hotel and Casino'},
{u'_id': u'Fiesta Henderson'},
{u'_id': u'Hilton Garden Inn'},
{u'_id': u'Crestwood Suites'},
{u'_id': u'Manor Suites'},
{u'_id': u'Tahiti Village'},
{u'_id': u'Micdrotel Inn'},
{u'_id': u'Mandarin Oriental'},
{u'_id': u'Aria Resort & Casino'},
{u'_id': u'Vdara'},
{u'_id': u'The Cromwell'},
{u'_id': u"Marriott's Grand Chateau"},
{u'_id': u'The Carriage House', u'phone': u'702.798.1020'},
{u'_id': u'The California Hotel and Casino'},
{},
{u'_id': u'Trump Hotel'},
{u'_id': u'Bellagio Hotel and Casino'},
{u'_id': u'Excalibur Hotel and Casino'},
{u'_id': u'Monte Carlo Hotel and Casino'},
{u'_id': u'New York New York Hotel and Casino'},
{u'_id': u'Caesars Hotel and Casino'},
{u'_id': u'The Quad Resort & Casino',
u'phone': u'(800) 634-6441'},
{u'_id': u'Venetian Hotel and Casino'},
{u'_id': u'Mandalay Bay Hotel and Casino'},
{u'_id': u'Flamingo Hotel and Casino'},
{u'_id': u'MGM Grand Hotel and Casino'},
{u'_id': u"Bally's Hotel and Casino"},
{u'_id': u'Rio All Suite Hotel and Casino'},
{u'_id': u'The Hotel at Mandalay Bay'},
{u'_id': u'Hooters'},
{u'_id': u"Harrah's Hotel and Casino"},
{u'_id': u'the D Las Vegas Hotel & Casino',
u'phone': u'(702) 388-2400'},
{u'_id': u'La Quinta Inn'},
{u'_id': u'Hard Rock Hotel and Casino'},
{u'_id': u'Wyndham Grand Desert'},
{u'_id': u'LVH Las Vegas Hotel & Casino',
u'phone': u'702-732-5111'},
{u'_id': u'Planet Hollywood Resort and Casino',
u'phone': u'+1 866 919 7472'},
{u'_id': u'The Westin Casuarina'},
{u'_id': u'Circus Circus Las Vegas Hotel and Casino'},
{u'_id': u'Stratosphere Hotel and Casino'},
{u'_id': u'J. W. Marriot Resort'},

{u'_id': u'Hampton Inn and Suites'},
{u'_id': u'The Palazzo'},
{},
{u'_id': u'Gold Strike'},
{u'_id': u'Spring Hill Suites'},
{u'_id': u'Town Place Suites'},
{u'_id': u'Wynn Hotel & Casino'},
{u'_id': u'The Grand'},
{u'_id': u'Hyatt Place'},
{u'_id': u'Super 8'},
{u'_id': u'Golden Gate Hotel and Casino'},
{u'_id': u'Palace Station'},
{},
{u'_id': u'Blue Moon'},
{u'_id': u'Eastside Cannery Casino Hotel'},
{u'_id': u'Hampton Inn'},
{u'_id': u'La Quinta Inn & Suites'},
{u'_id': u'Best Western McCarran Inn'},
{u'_id': u'SpringHill Suites by Marriott'}]}

Malls as amenity: 10

{u'ok': 1.0,
u'result': [{u'_id': u'Fashion Show Mall'},
{},
{u'_id': u'Las Vegas Outlet Center Annex'},
{u'_id': u'Crystals'},
{u'_id': u'Miracle Mile Shops'},
{},
{},
{},
{u'_id': u'Fashion Show Mall'},
{u'_id': u'Downtown Container Park'}]}

2. Top 5 contributors to the map
In addition, to find users who contribute to this map, I setup pipeline5 to query MongoDB and showed top 5 contributors as follow:

#top 5 contributors to the map

```
pipeline5 = [{"$match": {"created.user": {"$exists": 1}}},
{"$group": {"_id": "$created.user", "count": {"$sum": 1}}},
{"$sort": {"count": -1}},
{"$limit": 5}]
```

Outputs:
Top 5 contributors to the map: 5
{u'ok': 1.0,
u'result': [{u'_id': u'alimamo', u'count': 254892},
{u'_id': u'woodpeck_fixbot', u'count': 79690},

{u'_id': u'nmixter', u'count': 70150},
{u'_id': u'gMitchellD', u'count': 48266},
{u'_id': u'robgeb', u'count': 43484}]}

Conclusion: Other View about the Dataset

Based on my experience & understanding with exploring the OpenStreetMap data, I believe that the data structure is flexible enough to incorporate a vast multitude of user generated quantitative and qualitative data beyond that of simply defining a virtual map.

Further extending this open source project to include data such as user reviews of establishments, subjective areas of what bound a good and bad neighborhood, housing price data, school / hospitals reviews, walkability, quality of mass transit, and on would form a solid foundation of robust recommender systems. These recommender systems could aid users in anything from finding a new home or apartment to helping a user decide where to spend a weekend afternoon.

Unfortunately, it appears that, at least for the area of the world that I analyzed, the mapping data is far too incomplete to be able to implement such recommender systems. I believe that the OpenStreetMap project would greatly benefit from visualizing data on content generation within their maps. For example, a heat map layer could be overlayed on the map showing how frequently or how recently certain regions of the map have been updated. These map layers could help guide users towards areas of the map that need attention in order to help more fully complete the data set.