Washington DC OpenStreetMap Data Wrangling

Map area for this project is washington DC. The link is listed here: OSM XML 19MB

Our objective is: Audit, clean the OSM dataset, convert from XML to JSON format, insert to MongoDB and analyze insight within the data.

1. data audit

I checked the "k" value for each entry. I have a count of each of three tag categories in a dictionary: "lower", for tags that contain only lowercase letters and are valid, "lower_colon", for otherwise valid tags with a colon in their names, "problemchars", for tags with problematic characters.

```
{'lower': 445855, 'lower_colon': 793061, 'other': 34073, 'problemchars': 2}
```

Number of unique user:

```
1428
```

2. Problems

2.1 Street rename

Here is a problem encountered in this dataset come from the street name abbreviation inconsistency. Here is the plan of replace:

```
'Ave': 'Avenue', 'Blvd' : 'Boulevard', 'Dr' : 'Drive', 'Ln' : 'Lane', 'Pkwy' :
'Parkway', 'Rd' : 'Road',
'Rd.' : 'Road', 'St' : 'Street', 'street' :"Street", 'Ct' : "Court", 'Cir': "Circle", 'Cr' :
"Court",
'ave' : 'Avenue', 'Hwg' : 'Highway', 'Hwy' : 'Highway', 'Sq' : "Square"
```

2.2 Zip Code

Another problem is the zip code in this area, I found that some of zip codes are in different forms. I tried to handle those zip codes.

```
{None: set(['20005-1001',
            '20005-1009',
            '20005-1013',
            '20005-1015',
            '20005-1019',
            '20005-7700',
            '20011-6927',
            '20036-5305',
            '2005',
            '2011',
            '22202-1500',
            '22204-5360'])}
```

As we can see, there are 3 formats: the format of 5 digits are valid, all other than this: 4 digits and 5 digits - 4 digits need reformatting.

```
20005-1009 => 20005
20005-7700 => 20005
20005-1019 => 20005
20005-1001 => 20005
20036-5305 => 20036
2005 => None
20011-6927 => 20011
20005-1015 => 20005
22204-5360 => 22204
20005-1013 => 20005
22202-1500 => 22202
2011 => None
```

2.3 Phone number

There are many forms of phone number in this osm file, and I reformat all of them to 10 digits form such as:

```
202-429-0100 => 2024290100
+1 202 842 0414 => 2028420414
+12023884085 => 2023884085
202 722 4023 => 2027224023
202 525  5236 => 2025255236
+12022375555 => 2022375555
```

+1-202-827-8791 => 2028278791
(202) 216-0503 => 2022160503
+1 202-888-0050 => 2028880050

I removed all the space abnormal characters and country code for all numbers.

3 Convert XML to JSON

The basic plan of convert is from the follow rules:

- All attributes with "node" and "way" should be turned into regular key/value pairs, except: attributes in the "CREATED" array should be added under a key "created", 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 sub-level tag "k" value starts with "addr:", it should be added to a dictionary "address"
- If sub-level tag "k" value does not start with "addr:", but contains ":", you can process it same as any other "k" tags.
- If sub-level tag "k" value contains problematic characters, it should be ignored
- If there is a second ":" that separates the type/direction of a street, the tag should be ignored After all the data cleaning and data transformation are done, use last function "process_map" to convert the file from XML into JSON format.

The final JSON is in a format like:


```
{
   "website": "http://www.saint-ex.com",
   "cuisine": "Classic Bistro",
   "amenity": "bar",
   "name": "Cafe Saint Ex",
   "created": {
      "changeset": "41452792",
      "user": "Marion Barry",
      "version": "11",
      "uid": "408282",
      "timestamp": "2016-08-14T19:10:29Z"
   },
   "opening_hours": "Su 11:00-01:30; Mo 17:00-01:30; Tu-Th 11:00-01:30;
Fr-Sa 11:00-02:30",
   "wheelchair": "no",
   "atm": "yes",
```

```
    "pos": [
      38.9154205,
      -77.0317015
    ],
    "phone": "2022657839",
    "address": {
      "street": "14th Street Northwest",
      "housenumber": "1847",
      "postcode": null
    },
    "type": "node",
    "id": "60471261"
}
```


Them I upload this JSON format to mongoDB, and compare the new generated file with old file:

The DC.osm file size is 332.767148 mb.
The transformed JSON file size is 371.459312 mb.

Here is some query result summary from *.py file
Number of Documents:

```
>db.dc.find().count()
result : 1584162
```

Number of Documents with Street Addresses:

```
>db.dc.find({'address.street' : {'$exists' : 1}}).count()
result : 112565
```

Number of Documents with postcode:

```
>db.dc.find({'address.postcode' : {'$exists' : 1}}).count()
result : 54671
```

Number of Documents with phone:

```
>db.dc.find({'phone' : {'$exists' : 1}}).count()
result : 732
```

Number of Nodes and Ways:

```
>db.dc.find({'type':'node'}).count()
>db.dc.find({'type':'way'}).count()
```

Number of nodes: 1411526
Number of ways: 172587

Name of top 10 contributors:

```
>db.dc.aggregate([{"$group" : {"_id" : "$created.user", "count" : {"$sum" :
1}}},
                  {"$sort" : {"count" : - 1}}, {"$limit" : 10}])
```

The top 5 Most Referenced Nodes:

```
>db.dc.aggregate([{'$unwind': '$node_refs'}, {'$group': {'_id': '$node_refs',
'count': {'$sum': 1}}},
                  {'$sort': {'count': -1}}, {'$limit': 5}])
```

4. More data query with MongoDB

List of top 10 cuisine in DC:

```
>db.dc.aggregate([{"$match":{"amenity":{"$exists":1},"amenity":"restaurant",
}},
            {"$group":{"_id":{"Food":"$cuisine"},"count":{"$sum":1}}},
            {"$project" : {"_id":0,"Food":"$_id.Food","Count":"$count"}},
            {"$sort" : {"Count":-1}},
            {"$limit": 10}])
```

List of top 15 amenities in DC:

```
>db.dc.aggregate([{'$match': {'amenity': {'$exists': 1}}},
            {'$group': {'_id': '$amenity', 'count': {'$sum': 1}}},
            {'$sort': {'count': -1}},
            {'$limit': 15}])
```

List of top 10 Banks:

```
>db.dc.aggregate([{'$match': {'amenity': 'bank'}},
            {'$group': {'_id': '$name', 'count': {'$sum': 1}}},
```

```
                {'$sort': {'count': -1}},
                {'$limit': 10}])
```

List of top 10 restaurants:

```
>db.dc.aggregate([{'$match': {'amenity': 'restaurant'}},
                {'$group': {'_id': '$name', 'count': {'$sum': 1}}},
                {'$sort': {'count': -1}},
                {'$limit': 10}])
```

5. Conclusion

Some understanding of improve data quality of OSM:

When audit data, it was very clear that although there are minor error caused by human input, the dataset is fairly well-cleaned. Considering there might be hundreds of contributors on this map, human errors in this project will be very common here. I believe that a structured input form in needed so that everyone can input the same data format to reduce the chance of error or we can create a more robust script to clean the data regularly.

Potential cost of the implement structured input form:

There're few potential issues could you see that may arise from the implementation of this solution. One of which is the amount of effort to engineer all these processes and the cost of creating, auditing & maintaining these initiatives could be so overwhelm and require a dedicated team responsible for all these projects. Furthermore some information maybe not that easy to acquire, so it is necessary to design a wise algorithm to make judgement about whether the input format is valid.
For example, input phone number: it can be format as countrycode + 3 digits + 3 digits + 4 digits form, in this way, all the phone number are in same form.

6. Reference

http://wiki.openstreetmap.org/wiki/OSM_XML

MongoDB Importing XML to JSON Guide

Problem when converting xml file to JSON string