Sample Data Wrangling Project

OpenStreetMap Sample Project
Data Wrangling with MongoDB

Lekhraj Sharma

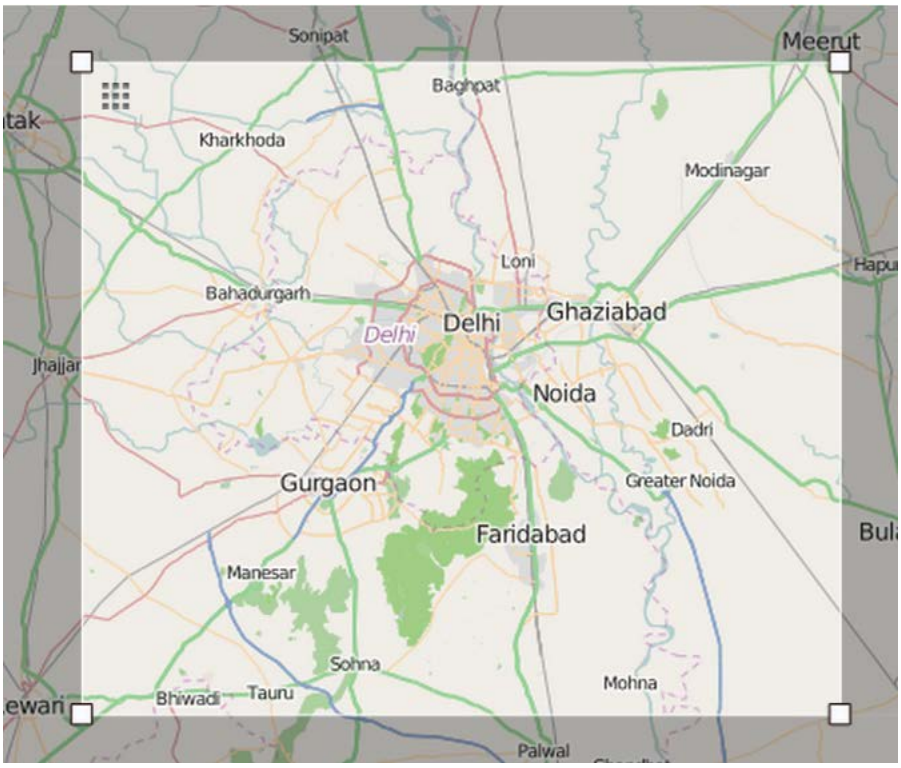
Map Area: New Delhi, New Delhi, India

Note:

- References consulted are given at the end of document.
- Map area: https://www.openstreetmap.org/export#map=9/28.5773/77.2119
- Map data: https://s3.amazonaws.com/metro-extracts.mapzen.com/new-delhi_india.osm.bz2
- Repository for this project containing code, data, output and other docs: https://github.com/DhammaWays/openstreetmaps

Table of Contents

0. Map: New Delhi, India



1. Problems Encountered in the Map

After downloading data set for New Delhi, India (see the map above) from metro extract, I ran several queries as described in exercise 6 of “Data Wrangling with MongoDB” course. Code and raw result files are located at my openstreetmaps repository at GitHub.

1.1 Familiarizing with data set

By just looking at fragments of downloaded OSM file, it is clear that dataset is very different than what you find with dataset of cities in US. Most of the focus is on marking amenities, landmarks, roads rather than proper street address for a location. It is also clear that folks marking these places have not strictly followed geographical boundary of actual city but lumped all nearby area of greater New Delhi city when tagging places and describing them.

1.2 Top Level Tags

The output of my mapparser.py (see below) indicates that it has close to half million nodes.

```
{'node': 502661, 'nd': 650577, 'bounds': 1, 'member': 13577, 'tag': 194160,
'relation': 2389, 'way': 87420, 'osm': 1}
```

1.3 Auditing and cleaning of street names

Auditing of street types with audit_street.py (see some extracts of output below, full output is in street.txt) indicates that we do not have much consistency in how street addresses have been added.

```
'Marg': set(['Abdul Gaffar Khan Marg',
             'Africa Avenue Marg',
             'Amrita Shergil Marg',
             'Aruna Asif Ali Marg',
            'Aurobindo Marg', …

'Nagar': set(['Adhyatmik Nagar',
              'Ansari Nagar',
              'Block B, Ashok Nagar Extension, New Ashok Nagar',..

Problematic street types:

'Delhi': set(['Block 35, Trilokpuri, New Delhi',
              'Ganesh Nagar I South, Pandav Nagar, New Delhi',
              'HPR School Main Road, Hira Colony, Siraspur, Delhi', …

Steet names corrected:
Arya Samaj Rd => Arya Samaj Road
```

1.4 Auditing and cleaning of city names

Auditing of city names with audit_city.py (see some extracts of output below, full output is in city.txt) indicates that we do not have much consistency in how city names have been added. This cleanup was added after MongoDB analysis (see section 1.9.3 Checking Postal Codes) where we discovered that inconsistently added city names were causing problems for us to analyze the postal zip code information.

```
'New Delhi': set(['Chanakyapuri, New Delhi',
                  'Pandav Nagar, New Delhi',
                  'Paschim Vihar, New Delhi',
                  'Pitam Pura, New Delhi',
                  'Pocket 6, Sector 9, Rohini, New Delhi',
                 'West Karawal Nagar, New Delhi'])

'Uttar Pradesh': set(['Noida , Uttar Pradesh', 'Noida, Uttar Pradesh'])


City names corrected:
 West Karawal Nagar, New Delhi => New Delhi
 Pocket 6, Sector 9, Rohini, New Delhi => New Delhi
 Paschim Vihar, New Delhi => New Delhi
 Pandav Nagar, New Delhi => New Delhi
 Pitam Pura, New Delhi => New Delhi
Chanakyapuri, New Delhi => New Delhi

 Noida , Uttar Pradesh => Noida
 Noida, Uttar Pradesh => Noida
```

1.5 Auditing tag keys

Auditing of tags with tag.py (see some extracts of output below, full output is in tags.txt) indicates that most keys are valid keys for MongoDB.

```
{'lower': 186001, 'lower_colon': 7987, 'other': 168, 'problemchars': 4}

Problematic Keys:
[{'k': 'Gaurav General Store', 'v': 'convenience'},
```

```
{'k': 'Aggarwal Sweets', 'v': 'fast_food'},
{'k': 'Ladies Readymade Garments', 'v': ''},
{'k': 'M.S. FLAT', 'v': ''}]
```

1.6 Unique Users

Finding number of unique users with users.py (see some extracts of output below, full output is in users_count.txt) indicates that a german user (oberaffe) is by far the leading contributor among 686 unique users with more than 250K entries (possibly via automation!).

```
Total number of unique users: 686
[('56597:Oberaffe', 269365),
 ("600918:n'garh", 83641),
 ('451671:Edolis', 52068),
 ('1306:PlaneMad', 16186),
 ('1292377:Nepolean', 14967),
 ('46622:roemcke', 14848),
 ('338611:marek kleciak', 14635),
 ('1292385:BalaK', 12166),
 ('1292408:Naveena', 12086),
```

1.7 Preparing for MongoDB analysis

Converted New Delhi OSM file to a more structured data model suitable for analysis with MongoDB with data.py. See some extracts of output below, full output is in new-delhi_india.osm.json file.

```
{"amenity": "school", "name": "Delhi Public School, Primary Section",
"created": {"changeset": "1428361", "user": "thevikas", "version": "2", "uid":
"17429", "timestamp": "2009-06-05T12:00:54Z"}, "pos": [28.4295352,
77.0525305], "type": "node", "id": "312101676"}

{"name": "Red Fort", "created": {"changeset": "27905598", "user": "PlaneMad",
"version": "2", "uid": "1306", "timestamp": "2015-01-04T10:14:22Z"},
"tourism": "attraction", "pos": [28.6552484, 77.2408328], "historic":
"archaeological_site", "place": "locality", "type": "node", "id":
"2705534625"}

{"amenity": "restaurant", "name": "Bukhara", "created": {"changeset":
"12245317", "user": "Oberaffe", "version": "1", "uid": "56597", "timestamp":
"2012-07-16T13:44:53Z"}, "pos": [28.6142041, 77.3715342], "type": "node",
"id": "1827530953"}

{"node_refs": ["913486694", "913486256", "913487014", "560646480",
"560646475"], "name": "Mahatma Gandhi Marg", "created": {"changeset":
"24652613", "user": "Oberaffe", "version": "27", "uid": "56597", "timestamp":
"2014-08-10T11:38:15Z"}, "oneway": "yes", "type": "way", "id": "23089147",
"highway": "primary"}
```

1.8 Creating MongoDB database

Loaded New Delhi OSM JSON data model file into a MongoDB collection “maps” with createdb.py.

1.9 Exploring data with MongoDB

Started to explore “maps” collection loaded in MongoDB with various queries (see the session.txt for these queries and their output). Some of these queries and observations are summarized below.

1.9.1 Checking geolocation

Wanted to explore if geo location are wrongly entered (i.e. not within bounds of our area New Delhi map: min_lat=281.183, min_lon=76.692, max_lat=28.969, max_lon=77.733). As can be seen from below MongoDB queries, most node entries seem to be fine.

```
 >db.maps.find({pos: {$geoWithin: {$box : [[28.183, 76.692], [28.969,
77.733]]}}}).count()
502661


>db.maps.find({type: "node"}).count()
502659
```

1.9.2 Checking type of documents

Wanted to explore if type of document was wrongly entered. As can be seen from following MongoDB queries that there are just very few handful entries that for whom type was wrongly entered.

```
 > db.maps.find({type: {$not: {$in: ["node", "way"]}}}).count()
 146

 > db.maps.find({type: {$not: {$in: ["node", "way"]}}}).limit(10)

{ "_id" : ObjectId("555c34e524a7ab09847dee00"), "name" : "Safdarjung
Airport", "is_in:country" : "India", "wikipedia:en" : "Safdarjung_Airport",
"created" : {
"changeset" : "4804409", "version" : "3", "user" : "PlaneMad", "timestamp" :
"2010-05-25T15:00:50Z", "uid" : "1306" }, "closest_town" : "New Delhi",
"pos" : [ 28.5849375, 77.206542 ], "ele" : "215", "icao" : "VIDD", "source"
: "wikipedia", "aeroway" : "aerodrome", "operator" : "Airports Authority of
India", "military" : "airfield", "type" : "Public", "id" : "410403385" }
…

 > db.maps.aggregate([{$group: {_id: "$type", count: {$sum: 1}}}])

 { "_id" : "bpundary", "count" : 2 }
 { "_id" : "boundary", "count" : 142 }
 { "_id" : "way", "count" : 87276 }
 { "_id" : "Public", "count" : 1 }
 { "_id" : "broad_palmate_leaves", "count" : 1 }
{ "_id" : "node", "count" : 502659 }
```

1.9.3 Checking postal codes

Wanted to explore if postal codes (called pin codes, zip codes as well) were correct. As can be seen from following MongoDB queries that most codes seem to be fine. Issue is more with not following a standard nomenclature in assigning them different cities within greater New Delhi area.

As following query points out there are only small percentage of entities having addresses where postal code has been entered.

```
> db.maps.find({"address.postcode": {$exists: true}}).count()
676

> db.maps.find({"address.postcode": {$exists: true}}).limit(5)

{ "_id" : ObjectId("555c34e324a7ab09847dde3f"), "website" :
"http://www.vivantabytaj.com/Ambassador-New Delhi/Overview.html", "name"
: "The Ambassador", "created" : { "changeset" : "20863949", "version" :
"2", "user" : "apm-wa", "timestamp" : "2014-03-02T13:08:07Z", "uid" :
"1960718" }, "type" : "node", "pos" : [ 28.6017391, 77.228846 ], "phone"
: "+91 11 6626 1000", "address" : { "city" : "New Delhi", "street" :
"Sujan Singh Park, Subramania Bharti Marg,Behind Khan Market",
"postcode" : "110003" }, "operator" : "Vivanta by Taj", "tourism" :
"hotel", "id" : "308894056" }
…
```

Postal code in Delhi start with 110 and in Noida 201. Checking the results of following query, postal code entered seems to be correct.

```
> db.maps.aggregate([{$group: {_id:"$address.city", zipcodes:{$addToSet:
"$address.postcode"}}}])

{ "_id" : "Noida, Uttar Pradesh", "zipcodes" : [ "201301" ] }
{ "_id" : "Pocket 6, Sector 9, Rohini, New Delhi", "zipcodes" : [ "110085"
] }
{ "_id" : "Gaziabad", "zipcodes" : [ "201001" ] }
{ "_id" : "Janakpuri", "zipcodes" : [ "110058" ] }
…
{ "_id" : "New Delhi", "zipcodes" : [ "110034", "110087", "110092",
"110065", "110005v",

 "110038", "110022", "110063", "110031", "110023", "110010", "110006",
 "110 001", "110044", "110019", "110015", "110020", "110037", "110046",
 "110014", "1100016", "110002", "110075", "110003", "110024", "110054",
 "110005", "110070", "110048", "110018",

"110016", "110025", "110085", "110011", "110021", "110096", "110017",
"110067", "110055", "110001", "110029", "1100049", "110058", "110078",
"110008" ] }
…
 { "_id" : "Noida", "zipcodes" : [ "201309", "201304", "201307",
 "201303", "201308", "2013010", "101301", "203135", "203202", "201010",
 "201301" ] }
```

Further digging in postal code, wanted to check if postal code have been correctly classified against cities in our map area. As following query indicates that there has not been much consistency on how data was logged against different cities. When we look at cities which has most number of postcodes, we find cities like “Greater Noida”, “Gurgaon” topping out “New Delhi” and “Delhi” area. This is not as per expectation as we should expect most postal codes should be belonging to “NewDelhi” area. We can see that while we do get reasonable numbers (289) if we add counts of both “New Delhi” and “Delhi”, it still falls short of cities in top of list. Looking closer we can see that fairly large number of addresses have not been tagged consistently in proper city format. Cities tagged as `"Hira Colony, Siraspur, Delhi"`, should have been tagged as “Delhi”. If we add up all these addresses and rightly tag them against New Delhi or Delhi, this city will be on top of our list as expected. Therefore dataset needs good amount of cleaning on address front.

```
> db.maps.aggregate([{$group: {_id:"$address.city", zipcodes:
{$addToSet: "$address.postcode"}, count:{$sum: 1}}}, {$sort: {count: -
1}}])

 { "_id" : "Greater Noida", "zipcodes" : [ "201310", "201306" ], "count"
 : 334 }
  { "_id" : "Gurgaon", "zipcodes" : [ "122003", "2242", "122004",
  "122016", "122002", "122017", "122001", "122018" ], "count" : 259 }
     { "_id" : "New Delhi", "zipcodes" : [ "110034", "110087", "110092",
     "110065", "110005v",
     "110038", "110022", "110063", "110031", "110023", "110010",
     "110006", "110 001", "110044", "110019", "110015", "110020",
     "110037", "110046", "110014", "1100016", "110002", "110075",
     "110003", "110024", "110054", "110005", "110070", "110048",
     "110018",
     "110016", "110025", "110085", "110011", "110021", "110096",
     "110017", "110067", "110055", "110001", "110029", "1100049",
     "110058", "110078", "110008" ], "count" : 164 }
   { "_id" : "Delhi", "zipcodes" : [ "110011", "110085", "110092",
   "110035", "110089", "201301", "110076", "110062", "110058", "110084",
   "201304", "110031v", "110052", "10089", "110096", "110093", "110001",
   "110087", "110006", "110002", "110091", "110054", "110067", "110032",
   "110020", "110034", "100006" ], "count" : 125 }
  { "_id" : "Noida", "zipcodes" : [ "201309", "201304", "201307",
  "201303", "201308", "2013010", "101301", "203135", "203202", "201010",
  "201301" ], "count" : 73 }
 { "_id" : "Hira Colony, Siraspur, Delhi", "zipcodes" : [  "110042" ],
 "count" : 50 }
 { "_id" : "Sector - 11, Rohini, Delhi", "zipcodes" : [ "110085" ],
 "count" : 15 }
  { "_id" : "Pandav Nagar, New Delhi", "zipcodes" : [ ], "count" : 14 }
...
```

So, these postal codes aren’t mapped consistently to their cities. See section 1.4 (Auditing/Cleaning of City names) on how we can clean-up the city names in our dataset.

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them. Please feel free to check session.txt for actual log of these queries and their output.

2.1 File Sizes

new-delhi_india.osm ..... 106 MB (compressed 10 MB)
new-delhi_india.osm.json .... 122 MB (compressed 10 MB)

2.2 Number of documents

```
>db.maps.count()
590081
```

2.3 Number of nodes

```
> db.maps.find({type: "node"}).count()
502659
```

2.4 Number of ways

```
> db.maps.find({type: "way"}).count()
87276
```

2.5 Number of unique users

```
> db.maps.distinct("created.user").length
677
```

2.6 Top 3 contributing users

```
> db.maps.aggregate([{$group:{_id:"$created.user", count:{$sum:1}}},
{$sort:{count:-1}}, {$limit:3}])

 { "_id" : "Oberaffe", "count" : 269265 }
 { "_id" : "n'garh", "count" : 81574 }
{ "_id" : "Edolis", "count" : 52010 }
```

2.7 Number of users appearing only up-to three times (having up-to 3 post)

Here “_id” in output represents how many times they have posted.

```
> db.maps.aggregate([{$group:{_id:"$created.user", count:{$sum:1}}}, {$group:
{_id:"$count", total_users:{$sum: 1}}}, {$sort: {_id: 1}}, {$limit: 3}])

 { "_id" : 1, "total_users" : 132 }
 { "_id" : 2, "total_users" : 62 }
{ "_id" : 3, "total_users" : 30 }

{ "_id" : "secondary", "count" : 1794 }
```

3. Additional Ideas

3.1 Additional Data Exploration Using MongoDB Queries

Following are additional data exploration using MongoDB queries. Please feel free to check session.txt for actual log of these queries and their output.

3.1.1 Number of top 10 amenities by count

No surprise that school top the list as capital city area is known as education hub.

```
 > db.maps.aggregate( [{$group: {_id:"$amenity", count:{$sum: 1}}}, {$sort:
 {count: -1}}, {$limit: 10}])
{ "_id" : null, "count" : 586793 }
{ "_id" : "school", "count" : 880 }
{ "_id" : "place_of_worship", "count" : 297 }
{ "_id" : "parking", "count" : 289 }
{ "_id" : "fuel", "count" : 197 }
{ "_id" : "hospital", "count" : 177 }
{ "_id" : "restaurant", "count" : 130 }
{ "_id" : "atm", "count" : 130 }
{ "_id" : "college", "count" : 124 }
{ "_id" : "bank", "count" : 100 }
```

3.1.2 Type of highways

No surprise that “residential” tops the type of highways as city has lot more residential roads than super national highways.

```
> db.maps.aggregate( [{$match: {type: "way"}}, {$group: {_id:"$highway",
count:{$sum: 1}}}, {$match: {count: {$gte: 1000}}}, {$sort: {count: -1}}] )

{ "_id" : "residential", "count" : 35686 }
{ "_id" : null, "count" : 23649 }
{ "_id" : "service", "count" : 12726 }
{ "_id" : "tertiary", "count" : 4022 }
{ "_id" : "living_street", "count" : 3636 }
{ "_id" : "unclassified", "count" : 2065 }
```

3.1.3 What are the religions practices in this area

Biggest religion is Hinduism but Muslims, Christians and Sikhs combined are almost same as number of hindus living in this area.

```
> db.maps.aggregate([{$match: {amenity: {$exists: 1},
amenity:"place_of_worship"}}, {$group: {_id:"$religion", count: {$sum:1}}},
{$match: {_id: {$ne: null}}}, {$sort: {count: -1}}])

{ "_id" : "hindu", "count" : 121 }
{ "_id" : "muslim", "count" : 47 }
{ "_id" : "christian", "count" : 33 }
{ "_id" : "sikh", "count" : 27 }
{ "_id" : "jain", "count" : 5 }
{ "_id" : "buddhist", "count" : 3 }
{ "_id" : "zoroastrian", "count" : 1 }
{ "_id" : "bahai", "count" : 1 }
```

3.1.4 Most popular cuisines

No surprise that “Indian” cuisine tops the list but “Chinese” and “Fast Food” is also popular!

```
>db.maps.aggregate([{$match: {amenity: {$exists: 1}, amenity:"restaurant"}},
{$group: {_id:"$cuisine", count: {$sum:1}}}, {$match: {_id: {$ne: null}}}, {$sort:
{count: -1}}, {$limit: 5}])

{ "_id" : "indian", "count" : 7 }
{ "_id" : "chinese", "count" : 3 }
{ "_id" : "vegetarian", "count" : 3 }
{ "_id" : "pizza", "count" : 3 }
{ "_id" : "burger", "count" : 2 }
```

3.2 Ideas to improve OpenStreetMap data

Following are some of ideas to improve quality of map data. The main challenge with any human entered data is accuracy and consistency as we have observed in analysis earlier. The other challenge is to create an incentive system for contributors to enter accurate/consistent data. Here are some of the ideas on how we can tackle these challenges in the context of OpenStreetMap.

3.2.1 Mobile App to assist in entering the data

Create a mobile app which allow user to auto tag a given location (using mobile GPS) and auto-fills the most required info (e.g. user name, user id, name of city, name of area, etc.) which user can override if they want to. It also lists down other fields which cannot be auto-filled for user to enter. It can do some basic check against all the fields right at entry time to nudge user to enter the data in right way. Other aspect of the mobile app is that it remembers past entries and auto-completes them if it can (e.g. zipcode if it is in same area). We could also add gamification to this app where we can award points and show the leaderboard of other contributors in their area.

3.2.2 Web widget for websites to encourage better quality of data for their own location

Create easy to use web widgets (for example that allow one to quickly see your location, get direction, search nearby) which an end user can embed in his/her website. We could initially tailor this web widget for prominent landmark buildings (schools, hospitals, police stations, restaurants, banks, etc.) so that they will be encouraged to correct the data for their own location if it is not accurate. To encourage easy correction we could provide an easy to use web based form to correct the info in relevant OpenStreetMap dataset.

3.2.3 Automated listing of problems and gamification of fixing them

Create automated test suites which identify possible problems in map data and list them as a challenge on website. Award points (based on difficulty level of problems to be corrected) to contributors who fix these problems. Advertise the leaders via a leaderboard. Award some special honorary badges (Diamond, Gold, Silver, etc.) which folks can put on their facebook/LinkedIn profiles. Announce some regular competitions where users (or teams) compete for top honors in fixing most number of problems (or most number of points). Give certificates/prizes for top teams.

4. Conclusion

After reviewing the data it is clear that greater New Delhi area data is not consistently entered. Tagging for various fields whether it be address, city, amenities, etc does not adhere to a well-defined norms. For example in address filed is just not the street address but entire postal address or just a landmark nearby! The data seems to focus on major landmarks and not accurate names. This lesson is quite reflected in google maps of this area where for direction it provides info on major landmarks on the way instead of relying on actual names of the road. Which is quite similar to what you will get if you ask a local person for direction! In spite of the consistency issue with the data, most queries deliver expected results, so data is fairly usable, law of statistics for a large sample at work here!

5. References

- https://www.openstreetmap.org
- https://mapzen.com/metro-extracts/
- https://wiki.openstreetmap.org/wiki/Main_Page
- http://docs.mongodb.org/manual/reference/
- http://www.w3schools.com/json/
- http://www.w3schools.com/xml/
- http://www.regexr.com/
- https://docs.python.org/2/
- http://stackoverflow.com/