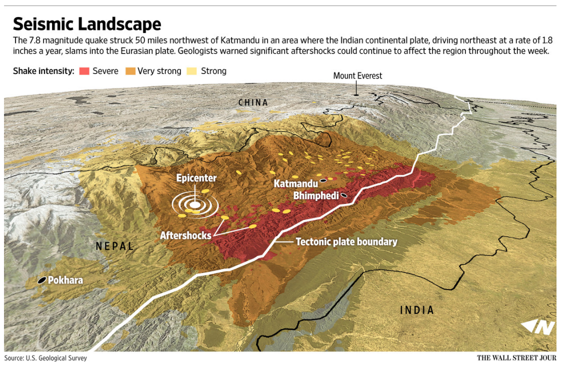
Open Street Map Project Report

Map Area: Kathmandu, Nepal
Author: Ian Xuan Xiao
Date: May 18, 2015

Introduction

Two devastating earthquakes hit Nepal on April 25th and May 12th 2015 respectively. The 7.8 and 5.7 magnitude quakes with epicenters at northwest and east of Kathmandu cause deadly damage in the area.


Seismic Landscape
The 7.8 magnitude quake struck 50 miles northwest of Katmandu in an area where the Indian continental plate, driving northeast at a rate of 1.8 inches a year, slams into the Eurasian plate. Geologists warned significant aftershocks could continue to affect the region throughout the week.
Shake intensity: Severe Very strong Strong
Mount Everest
CHINA
Epicenter
Katmandu
Bhimphedi
Aftershocks
Tectonic plate boundary
NEPAL
Pokhara
INDIA
Source: U.S. Geological Survey
THE WALL STREET JOUR

SHAKE INTENSITY
Moderate
Strong
Very strong
50 miles
50 km
CHINA
NEPAL
Epicenter of April 25 quake
Epicenter of Tuesday's largest quake
EURASIAN PLATE
Pokhara
MOUNT EVEREST
INDIAN PLATE
Kathmandu
Aftershocks
INDIA
Source: U.S. Geological Survey
THE WALL STREET JOURNAL.

Figures: *Illustration of epicenter of first and second earthquake; first quake on left, second quake on right;* ***Source***: *Wall Street Journal*

I took the opportunity of this project to investigate the quality of Kathmandu map data on Open Street Map (OSM) with the hope to improve the local data and use the finding to support local rescue and medical teams.

Beyond cleaning the data in my local MongoDB, a list of medical and shelter locations such as hospital, clinic, pharmacy, place of worship, and drinking water, etc., has been submitted to a responding organization (http://quakemap.org/)

In addition, data quality summaries have been sent to Humanitarian OpenStreetMap Team (http://hotosm.org/) to help prioritize the data improvement effort.

Data Problems

Missing Locations to Important Amenities

Significant amount of location information is missing in important facilities such as hospital, clinics, and pharmacies, etc., in Kathmandu area. See MongoDB query section for detailed query pipelines.

Type of Amenity	**Total Counts in Dataset**	**Total Counts with Location**	**Total Counts without Location**	**% Missing Location***
Hospital	254	72	182	71.7%
Clinic	189	85	104	55.0%
Pharmacy	217	194	22	10.1%
Drinking Water	193	183	7	3.63%
Place of Worship	1028	627	400	38.9%
Health Post	88	1	87	98%
Shelter	99	33	66	66.7%
Community Center	54	25	29	53.7%

**Note: the summation of total counts with location and missing location may not equal to total counts in dataset for a given amenity type due to query condition applied, such as type of tag; the percentage is calculated based on total counts in dataset for approximation; see MongoDB Query section for details*

Cleansing Approach: extensive insight to local environment is required to improve this data; report has been submitted to local responding group for further investigation and feedback.

Street Data Issues

1) **Street name with typos and spelling inconsistency**

For example:
‘Gali'
‘Galli’ -> correct word: small road that cannot access by vehicles
'marg'
'Marg-1'
'marg-1'
'marga'

Cleansing Approach: identify the correct spelling and standardize

2) **Street names with inconsistent capitalization**

'Chowk'
'chowk'

Cleansing Approach: Use capital letter, such as “Chowk” and “Road”

3) **Missing street names**

'1': set(['Hospital Marga 1', 'Pragati Marga 1']),

```
'2': set(['Kumari Galli 2']),
'4': set(['Sudal ward no - 4']),
'44900': set(['Khadga Bhanjyang 44900']),
'5': set(['Lalitpur 5']),
'Nepal': set(['Devighat Khadga Bhanjyang 44900 Nepal',
              'Lainchaur Kathmandu Nepal',
              'Paknajol, Kathmandu, Nepal']),
```

Cleansing approach: pertain these records for feedback and future edition based on local knowledge

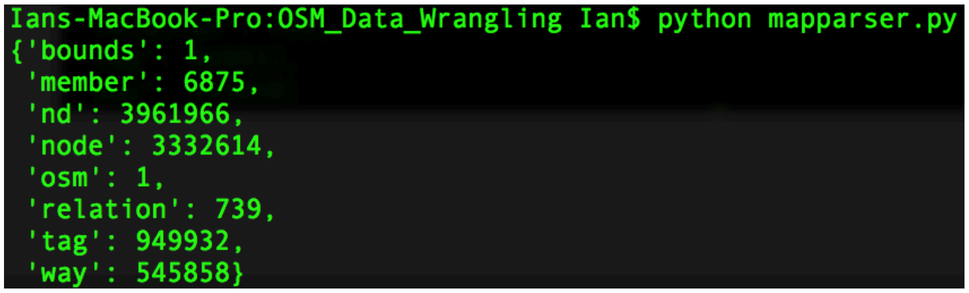
4) **Street abbreviation standardization**

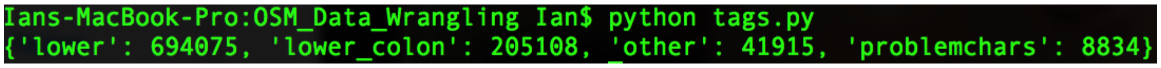
```
'ROAD': set(['RING ROAD']),
'Rd': set(['Amarnibas Rd',
           'Bijulibazar Rd',
           'British-Indian Embassy Rd',
           'Krishi Rd',
           'Madan Bhandari Rd']),
'Rd.': set(['Phulbari - Bhakunde Rd.'])
```

Cleansing approach: identify all relevant abbreviation and standardize

Issues in Tags Nested in Node and Way

Using tag type and its profile to identify and focus on areas of improvements,


```
Ians-MacBook-Pro:OSM_Data_Wrangling Ian$ python mapparser.py
{'bounds': 1,
 'member': 6875,
 'nd': 3961966,
 'node': 3332614,
 'osm': 1,
 'relation': 739,
 'tag': 949932,
 'way': 545858}
```


```
Ians-MacBook-Pro:OSM_Data_Wrangling Ian$ python tags.py
{'lower': 694075, 'lower_colon': 205108, 'other': 41915, 'problemchars': 8834}
```

The following statistics helps to develop cleansing approach

Type of Tags	**Definition**	**% of total of 949932 tags**
Lower	Normal tags	73%
Lower_colon	Tags with colon	21%
Problemchars	Tags with special characters	0.93%
Other	Tags do not fit in buckets above	4.4%

Cleansing approach: To pertain potential useful information, I removed colons in tags and include the cleansed tags as the keys in data structure when importing to MongoDB

Data Overview

Data of Retrieval: May 13, 2015, 8:43:33PM EST
Initial XML File Size: 718.8MB
Final JSON File Size: 840.4MB

See detail data profile section below.

MongoDB Queries

Number of distinct users

```
> len(db.distinct("created.user"))
> 4304
```

Number of distinct data sources

```
> len(db.distinct("source"))
> 196
```

Number of distinct amenity
```
> len(db.distinct("amenity"))
> 126
```

The following queries aimed to achieve the following,

1) Identify top emergency amenities related to medical, shelter, and assistance
2) Extract all emergency locations with detail longitude and latitude information
3) Extract all elements with location information missing to help priority data improvement effort
4) Provide data quality summaries of emergency location

Top 20 amenity sorted by occurrence

Query:
```
  pipeline =[{"$match":{"amenity":{"$exists":1}}},
          {"$group": {"_id": "$amenity", "count":{"$sum":1}}},
          {"$sort": {"count":-1}},
          {"$limit":100}
```

Output:
Only the ones with significant number of records are listed; yellow highlight are the ones in scope of my analysis)

```
{u'count': 2209, u'_id': u'school'}
{u'count': 1028, u'_id': u'place_of_worship'}
{u'count': 533, u'_id': u'restaurant'}
{u'count': 370, u'_id': u'bank'}
{u'count': 356, u'_id': u'college'}
{u'count': 257, u'_id': u'cafe'}
{u'count': 254, u'_id': u'hospital'}
{u'count': 217, u'_id': u'pharmacy'}
{u'count': 193, u'_id': u'drinking_water'}
{u'count': 189, u'_id': u'clinic'}
{u'count': 143, u'_id': u'kindergarten'}
{u'count': 136, u'_id': u'fast_food'}
{u'count': 117, u'_id': u'parking'}
{u'count': 113, u'_id': u'public_building'}
{u'count': 110, u'_id': u'atm'}
{u'count': 99, u'_id': u'shelter'}
{u'count': 94, u'_id': u'fuel'}
{u'count': 88, u'_id': u'health_post'}
{u'count': 87, u'_id': u'police'}
{u'count': 66, u'_id': u'toilets'}
{u'count': 57, u'_id': u'bus_station'}
{u'count': 54, u'_id': u'community_centre'}
{u'count': 40, u'_id': u'office'}
```

All emergency facilities with location information

Query:

```
    pipeline =[{"$match": {"amenity":{"$exists":1},
                           "pos":{"$exists":1}}},
                       {"$match": {"amenity":{"$in":["hospital","clinic","pharmacy",
                        "place_of_worship","drinking_water", "shelter", "health_post",
                        "community_centre"]}}},
                       {"$project": {"amenity":"$amenity","Name":"$name", "location":"$pos"}}
```

Output is too long to be included. Please see *shelter.json* in the output folder.

All emergency facilities without location information

```
     pipeline =[{"$match":{"amenity":{"$exists":1},
                           "pos":{"$exists":0},
                           "type":{"$in":["node", "way"]}}},
    {"$match": {"amenity":{"$in":["hospital","clinic","pharmacy",
     "place_of_worship","drinking_water", "shelter", "health_post",
     "community_centre"]}}}
        ]
```

Output is too long to be included; Please see *missing_location.json* in the output folder.

Data profile of emergency facilities with location information

Query:

```
shelter_profile_pipeline =[{"$match": {"amenity":{"$exists":1},
                                       "pos":{"$exists":1},
                                       "type":{"$in":["node", "way"]}}},
                          {"$match": {"amenity":{"$in":["hospital","clinic","pharmacy",
                                      "place_of_worship","drinking_water", "shelter",
                                      "health_post", "community_centre"]}}},
                          {"$group": {"_id": "$amenity", "count":{"$sum":1}}},
                          {"$sort":{"count":-1}} ]
```

Output:

```
{u'count': 627, u'_id': u'place_of_worship'}
{u'count': 194, u'_id': u'pharmacy'}
{u'count': 183, u'_id': u'drinking_water'}
{u'count': 85, u'_id': u'clinic'}
{u'count': 72, u'_id': u'hospital'}
{u'count': 33, u'_id': u'shelter'}
{u'count': 25, u'_id': u'community_centre'}
{u'count': 1, u'_id': u'health_post'}
```

Data profile of emergency facilities without location information

Query:

```
missing_profile_pipeline =[{"$match": {"amenity":{"$exists":1},
                                       "pos":{"$exists":0},
                                       "type":{"$in":["node", "way"]}}},
                          {"$match": {"amenity":{"$in":["hospital","clinic","pharmacy",
                                      "place_of_worship","drinking_water", "shelter",
                                      "health_post", "community_centre"]}}},
                          {"$group": {"_id": "$amenity", "count":{"$sum":1}}},
                          {"$sort":{"count":-1}} ]
```

Output:

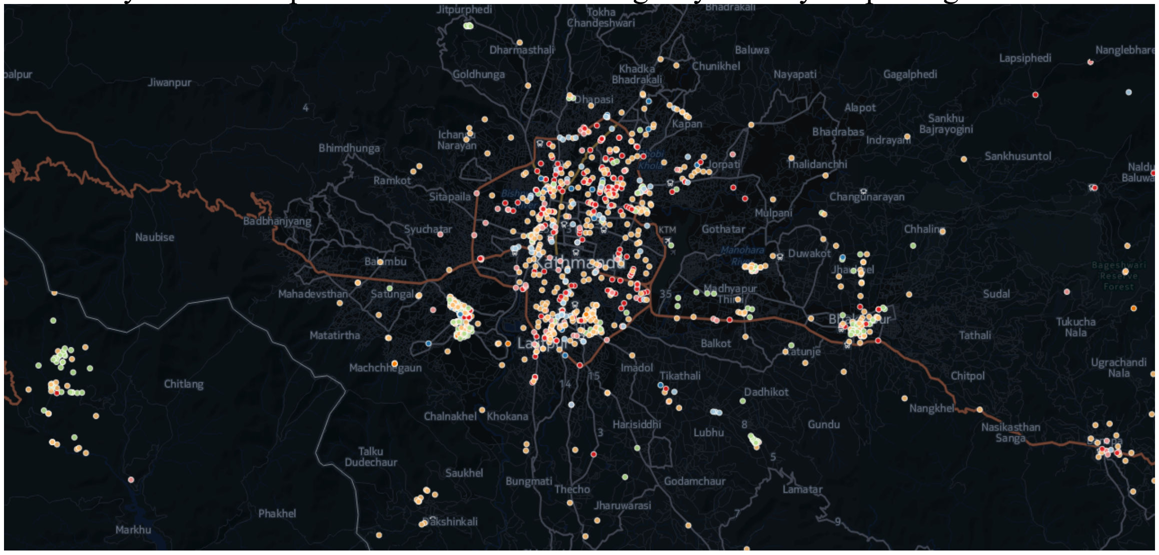
```
{u'count': 400, u'_id': u'place_of_worship'}
{u'count': 182, u'_id': u'hospital'}
{u'count': 104, u'_id': u'clinic'}
{u'count': 87, u'_id': u'health_post'}
{u'count': 66, u'_id': u'shelter'}
{u'count': 29, u'_id': u'community_centre'}
{u'count': 22, u'_id': u'pharmacy'}
{u'count': 7, u'_id': u'drinking_water'}
```

Other Idea about the Data Set

In addition to submitting my query outputs to local responding groups and OSM teams, I want to visualize the locations of all emergency facilities using cloud-based mapping tool such as Cartodb (http://cartodb.com/). This will help to further validate the accuracy of the facility locations and help responding teams to plan their rescue, medical, and re-build activities.

There are various challenges that I foresee in implementing the visualization. Most importantly, the location data should be live with updates from the OSM or local field teams. To achieve this, I need to develop regular extract from OSM or MapZen, re-run the MongoDB queries, and feed the new locations to the online mapping tool. Secondly, the visualization should be delivered to the field teams in a consumable and integrated way in order to make impact. Further investigations related to Cartodb API, cross platform integration, and mobile application compatibility are required.

This is my initial set up for the Kathmandu Emergency Facility Map using CartoDB.



Here is the link to the fully interactive map: http://cdb.io/1ecknpg
By the initial glance, the locations of the facilities seem to be accurate because they are mapped to the expected area and cluster around highly populated areas.