# Wrangling OpenStreetMap Data

Dataset: Bangalore City (size of osm: 649.3MB)

## Brief introduction the Map Area

Bengaluru, Karnataka, INDIA is my hometown, and is a with a lot of eateries, startups and Busy Streets. Hope my project helps in the contribution towards OSM data of Bengaluru

http://www.openstreetmap.org/export#map=11/12.9791/77.5916
https://mapzen.com/data/metro-extracts/metro/bengaluru_india/

## Problems Encountered

Few problems that occurred

1. Abbreviations in road names
2. Entire address in road name
3. Local Language problem

---

### Abbreviations of road names

The dataset was different from traditional osm files of large cities. Most of the types of node or way tags were “name” and of type “regular”. So fixing street names was done in a slightly different way. We just looked through every <way> and looked through the content of it to determine if it was a road. And on finding it was a road, we cleaned the data to fix bad abbreviations to the street names.

```
def value_fixer(value):
#     Make all upper/lower case of 'road|cross|main' to Road
    re_road = re.compile(re.escape('road'), re.IGNORECASE)
    re_main = re.compile(re.escape('main'), re.IGNORECASE)
    re_cross = re.compile(re.escape('cross'), re.IGNORECASE)
    if re.search('road', value, re.IGNORECASE):
        value = re_road.sub(‘ Road',value)
    if re.search('cross', value, re.IGNORECASE):
        value = re_road.sub('Cross',value)
    if re.search('main', value, re.IGNORECASE):
        value = re_road.sub('Main',value)
```

```
#     Fix Rd or Rd. to Road
#     Added a space so as to avoid considering 'rd' as a part of '3rd'
and only taking abc rd.
    re_rd_period = re.compile(re.escape(' rd\.'), re.IGNORECASE)
    re_rd = re.compile(re.escape(' rd'), re.IGNORECASE)
    if re.search(' rd\.', value, re.IGNORECASE):
        value = re_rd_period.sub(' Road', value)
    elif re.search(' rd', value, re.IGNORECASE):
        value = re_rd.sub(' Road', value)
```

Similarly cross and main were also fixed.

CLEANED VALUE EXAMPLES
Old value: 100 feet Rd, EjipuraBangalore, Karnataka, India
New value: 100 feet Road, EjipuraBangalore, Karnataka, India

Old value: 80 Feet Peripheral Rd, Koramangala 6 Block, KoramangalaBengaluru, Karnataka, India
New value: 80 Feet Peripheral Road, Koramangala 6 Block, KoramangalaBengaluru, Karnataka, India

Old value: Tuasi theater rd, 3rd cross, Marathalli east
New value: Tuasi theater Road, 3rd cross, Marathalli east

Old value: 7th Cross Rd, BTM Layout 2Bengaluru, Karnataka
New value: 7th Cross Road, BTM Layout 2Bengaluru, Karnataka

Old value: 7th Cross Rd, BTM Layout 2Bengaluru, Karnataka
New value: 7th Cross Road, BTM Layout 2Bengaluru, Karnataka

---

## Entire address in road name

Sometimes the entire road name given in the format “15 Main, Area_Name” so, any text after main or road is removed.

```
if 'Road' in value:
    value=value[0:value.index('Road')+4]
```

```
      if 'Road' in value:
          value=value[0:value.index(‘Road')+4]
```

CLEANED VALUE EXAMPLE

```
Old value: Outer ring Road, Bellandur
New value: Outer ring Road
Old value: Outer ring Road, Bellandur
New value: Outer ring Road
Old value: Outer ring Road, Bellandur village
New value: Outer ring Road
Old value: Sundari Armadale, Whitefield Main Road,
Whitefield
New value: Sundari Armadale, Whitefield Main Road
old value: Crescent Road,Nehru Nagar,Gandhi Nagar
new value: Crescent Road
```

---

## Local Language Problem

Even if I had to put type, the data also contained data written in the local language 'Kannada', so the key of tags became 'name:kn' Now in such a case, would mean the key was 'name' and type was 'kn'. What was required is key:'name:kn' and type: 'regular', to make it uniform with the rows which were written in english. i.e. key:'name' and 'type' english. This problem was solved programmatically by ignoring the ':kn' part of the key.

EXAMPLE

1. <tag k='name:kn' v='ಂನೇ ಸಿ ಮುಖ್ಯ ರಸ್ತೆ '>

type: regular
key: name:kn
value: ಂನೇ ಸಿ ಮುಖ್ಯ ರಸ್ತೆ

INSTEAD OF
type: name
key: kn
value: ಂನೇ ಸಿ ಮುಖ್ಯ ರಸ್ತೆ

2. <tag k='name:kn' v='ನ್ಯಾಶನಲ್ ಪ್ರೌಡ ಶಾಲಾ ರಸ್ತೆ '>

type: regular
key: name:kn
value: ನ್ಯಾಶನಲ್ ಪ್ರೌಡ ಶಾಲಾ ರಸ್ತೆ

INSTEAD OF
type: name
key: kn
value: ನ್ಯಾಶನಲ್ ಪ್ರೌಡ ಶಾಲಾ ರಸ್ತೆ

## CONCLUSION

On getting the data in the right format, the data was validated, and then uploaded to the database. While uploading, I couldn't convert to string. So I had to convert it to 'utf-8' to be able to upload the documents.

# DATA OVERVIEW

## FILE SIZES

Size of each CSV file it was retreived from¶

1. Size of node.csv file: 241.3 MB
2. Size of node_tags.csv file: 3.7 MB
3. Size of way.csv file: 40.3 MB
4. Size of way_nodes.csv file: 85.8 MB
5. Size of way_tags.csv file: 24.1 MB

## NUMBER OF ROWS

```
sqlite> SELECT COUNT(*) FROM row_type;
```

1. Number of nodes: 2882959
2. Number of node_tags: 93243
3. Number of ways: 660784
4. Number of way_nodes: 3576371
5. Number of way_tags: 723631

## Number of Rows in the local language 'Kannada'

```
sqlite> SELECT COUNT(*) FROM node_tag WHERE key LIKE '%:kn%';
```

RESULT

1. In node tag: 4444
2. in way tag: 8109

## Number of unique users

```
sqlite> SELECT COUNT(DISTINCT(e.uid)) FROM (SELECT uid FROM node UNION
ALL SELECT uid FROM way) e;
```

RESULT 1999

---

### Highest contributing users

```
sqlite> SELECT e.user, COUNT(*) as num FROM (SELECT user FROM node
UNION ALL SELECT user FROM way) e GROUP BY e.user ORDER BY num DESC
LIMIT 10;
```

RESULT: (u'jasvinderkaur', 124900)
(u'akhilsai', 118687)
(u'premkumar', 115884)
(u'saikumar', 114996)
(u'shekarn', 98118)
(u'PlaneMad', 94732)
(u'vamshikrishna', 94275)
(u'himalay', 88246)
(u'himabindhu', 86844)
(u'sdivya', 84998)

## Additional Data Exploration

---

### Popular Amenities

```
sqlite> SELECT value,count(*) FROM node_tags WHERE key LIKE
'%amenity%' GROUP BY value ORDER BY count(*) DESC LIMIT 10;
```

RESULT
(u'restaurant', 1697)
(u'atm', 799)
(u'bank', 743)
(u'place_of_worship', 701)
(u'pharmacy', 553)
(u'fast_food', 515)

(u'hospital', 454)
(u'school', 371)
(u'cafe', 350)
(u'fuel', 282)

---

## INVESTIGATING RESTAURANT, CAFE, FAST FOOD JOINTS - Cuisine

```
sqlite> SELECT value,count(*) as quantity FROM node_tags,(SELECT
DISTINCT(id) FROM node_tags WHERE value IN
('restaurant','cafe','fast_food')) as foodnodes ON
node_tags.id=foodnodes.id WHERE key IN ('cuisine') GROUP BY value
ORDER BY quantity DESC LIMIT 10;
```

RESULT
(u'regional', 368)
(u'indian', 292)
(u'pizza', 90)
(u'vegetarian', 89)
(u'chinese', 78)
(u'ice_cream', 52)
(u'coffee_shop', 50)
(u'burger', 45)
(u'international', 31)
(u'italian', 29)

---

## NUMBER OF ROADS IN BANGALORE

```
SELECT count(*) FROM way_tags WHERE value LIKE ‘%Road%’
```

RESULT: 10325

---

## NODE ID WHERE HIGHEST NUMBER OF WAYS PASS THROUGH

```
node_through_which_highest_way_pass = "(SELECT value,count(*) as
quantity FROM node_tags,(SELECT DISTINCT(id) FROM node_tags WHERE
value IN ('restaurant','cafe','fast_food')) as foodnodes ON
node_tags.id=foodnodes.id WHERE key IN ('cuisine') GROUP BY value
ORDER BY quantity DESC LIMIT 10”
```

```
info_about_that_node = "SELECT id FROM
node,"+node_through_which_highest_way_pass+" ON
busy_nodes.node_id=node.id WHERE node.id=busy_nodes.node_id "
```

```
result = c.execute(info_about_that_node)
op=[]
for row in result:
    op.append(row[0])
print op
```

```
OUTPUT: [3676386504, 3756817769, 3756817774, 3676386503, 3750785900,
3751654300, 3756817768, 3676374899, 3676374909, 3750785901]
```

# Other ideas about the dataset

---

## VIDHAN SOUDA - The all the most happening nodes are around this area

There were no information on these nodes, so I went through google maps entering these lat,lons. Found something interesting. All these nodes are a part of or few minutes away from 'Vidhan Souda' which is the state legislature building of karnataka. Which makes sense. There might be a lot of ways - including the walking paths, buildings, walls, metro, roads that pass through it, roads that go around it, service roads, etc. Information can be added about this place, and this can be done through various means including,

- Providing right incentives to add information
- Checking other adjacent nodes for appropriate information, and adding information programmatically

This is beneficial as the location is really important. But there are a lot of empty valued nodes. There might be few issues such as naming the tags accurately.