Openstreetmap Data Case Study

Map Area: Bengaluru Urban, India

1. Data Auditing:

i) **Map parser:** After the basic analysis of OSM data, the tags which i came across and the number of occurrences are as follows:

bounds	1
member	6617
nd	3577785
node	2884367
osm	1
relation	133
tag	820734
way	660914

ii) **Tag Types:** For all the elements that contains tags, those are checked for the 'k value' and in case of problems it is report below:

tag category	count	desc.
lower	779573	for tags that contain only lowercase alphanumeric letters and are valid.
lower_colon	40038	for otherwise valid alphanumeric tags with a colon in their names.
upper	46	for otherwise valid upper case letters tags.
problemchars	2	for tags with problematic characters
other	1075	for other tags that do not fall into the other three

		categories

I came across 2 tags with a problematic character, and 1075 that don't belong to any of the categories.

Problematic Character:
ರಾಜಗೋಪಾಲ ನಗರ ರಸ್ತೆ
71st Cross Road
(Data in a different language and data starting from number)

Others:
service:bicycle:retail
name:iso15919
sidewalk:both:kerb
ref_1
**and many others*

They either mix upper with lowercase letters and numeric, include more than one colon, or written in Kannada.

2. Problems Encountered in the Map:

To narrow it down, i decided to audit the street names only in my OSM dataset. I have printed the streets name excluding the common street ending tag like road, street, Avenue etc.
Finally i ended up with many inconsistencies which i kept adding in the Mapping table manually as below:

```
mapping = {   "St": "Street",
              "St.": "Street",
              "Ave": "Avenue",
              "Rd.": "Road",
              "Road": "Road",
              "road": "Road",
              "cross": "Cross",
              "ROad": "Road"        }
```

Inconsistent Street Naming:
The problems can be listed down to many categories:

i) **Problems with Kannada Street names:**

Translation API is required to verify the information

ii) **Problem with English Street names:**

- Abbreviations
  - St. , St -> Street
  - Ave -> Avenue

- Lower Case
  3 rd Cross Ramaswamy layout New byappanhalli extn bangalor560038
- St/Rd keywords are in the middle of the street name
  Begur-Koppa Rd,
- Inconsistent lower and upper letter cases
  Margosa Ave,green glen layout,bellandur
- Missing keywords *street/road* at the end of the name
  6 main Basaveshwarnagar

Inconsistent Postal Codes:

- Spaces in postal codes
  560 028 -> 560028
- Extra 0 in postal codes
  5600011 -> 560011
- Other postal codes
  Invalid *-> NA*

3. Data Cleaning

In this section the street names are detected and corrected using regular expressions. Since the data is very big (around 1 GB) and which is beyond the scope. Hence, i have only worked on the street names and corrected those only.

Using the code “bengaluru_india.osm.json” i have updated the English Street names for a sample either by mapping them to better written names, capitalizing the first letter based on below mapping table.

```
mapping = {   "St": "Street",
              "St.": "Street",
              "Ave": "Avenue",
              "Rd.": "Road",
              "Road": "Road",
              "road": "Road",
              "cross": "Cross",
              "ROad": "Road"        }
```

4. Data Overview:

This section gives the general stats about the data, and the random queries done to produce some of the relevant informations which can be used to take this project for further analysis. The scope of this project is to make a JSON type database for a sample and correct the anomalies present.

For the sample output created, the counts are less than what is present in OSM file.

Some of the other informations are listed for the sample taken to produce the JSON database:

i) **Number of nodes**: 44791

ii) **Number of tags**: 5208

For counting above tags following code is used:

```
for event,elem in ET.iterparse(osm_file):
        current = elem.tag
        counts[current] += 1
```

iii) **Number of unique users**: 446

```
        unique_user = defaultdict(int)
        uid=node["created"]["uid"]
        unique_user[uid] += 1
```

iv) **Most popular Cuisine**: pizza

```
if elem.tag == 'tag':
        if (key_k=="cuisine"):
            cuisine[key_v] += 1

print max(zip(cuisine.values(),cuisine.keys()))
```

v) **Most followed religion**: hindu

```
if elem.tag == 'tag':
         elif(key_k=="religion"):
            religion[key_v] += 1

print max(zip(religion.values(),religion.keys()))
```

5. Additional Ideas

Benefits:
There are a lot of information present which can be utilized in extracting relevant information. Some of them i have listed below which can be extracted and used:

1. Type of crisines

```
if elem.tag == 'tag':
        if (key_k=="cuisine"):
            cuisine[key_v] += 1
```

{'chinese': 2, 'indian;chinese;regional': 1, 'punjabi': 1, 'regional': 1, 'South_Indian': 1, 'burger': 2, 'american': 1, 'indian': 5, 'ice_cream': 2, 'pizza': 5})

These can be filtered more by categorising multiple cuisine restaurant in right way.

2. Type of religions

```
if elem.tag == 'tag':
        elif(key_k=="religion"):
            religion[key_v] += 1
```

{'hindu': 22, 'muslim': 5, 'christian': 10, 'jain': 2})

This data can be further analysed to figure out which region has majority of particular region to relate with the number of restaurant present with particular cuisine to get the correlation.

Anticipated Problems:

- There are problem in linking API for language translation and then verifying those data. The code will become bit complex. Plus it will require bit of Machine Learning to implement it which is not in the scope of this module.

- Data available is pretty neat. Still there can be multiple issues with respect to data entry. Validation of entered data required secondary check from trusted sources.

6. Conclusion

Dealing with Bengaluru dataset was tough because of many inconsistencies in data entry. At some place there was language change, street names didn't follow any rule for entries. There were several abbreviation for a same word which made this task very tedious, it required manual check for those abbreviations. However, the sample data converted in

JSON is cleaned properly. Cleaning whole Bengaluru dataset is still not in the scope of this project.

7. References

- https://mapzen.com/data/metro-extracts/metro/bengaluru_india/
  For XML download