

This repository


Search


Pull requests

Issues

Marketplace

Gist



 shubh12et1062 / 3-Wrangle-OpenStreetMap-Data

Unwatch

1

Star

0

Fork

0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Settings


Insights

Branch: master

3-Wrangle-OpenStreetMap-Data / README.md

Find file

Copy path

 shubh12et1062 Update README.md

b2c65b7 just now


1 contributor


212 lines (185 sloc) 6.02 KB


Raw

Blame

History







# Project 3: OpenStreetMap Data Wrangling with SQL

**Name:** Subham Biswas

**Map Area:** I have choosen Mumbai as i am living here.

- Location: Mubai, India
- [OpenStreetMap URL](#)
- [MapZen URL](#)

## 1. Data Audit

###Unique Tags Looking at the XML file, I found that it uses different types of tags. So, I parse the Mumbai,India dataset using ElementTree and count number of the unique tags. tags.py is used to count the numbers of unique tags.

```
##'member': 1195,
##'nd': 234500,
##'node': 204412,
##'osm': 1,
##'relation': 399,
##'tag': 38809,
##'way': 28327
```

### Patterns in the Tags

The "k" value of each tag contain different patterns. Using tags.py , I created 3 regular expressions to check for certain patterns in the tags. I have counted each of four tag categories.

- "lower" : 37303 , for tags that contain only lowercase letters and are valid,
- "lower\_colon" : 1449 , for otherwise valid tags with a colon in their names,
- "problemchars" : 55 , for tags with problematic characters, and
- "other" : 2 , for other tags that do not fall into the other three categories.

## 2. Problems Encountered in the Map

###Street address inconsistencies The main problem we encountered in the dataset is the street name inconsistencies. Below is the old name corrected with the better name. Using audit\_street.py , we updated the names.

```
* **Abbreviations**
* `Rd -> Road`
* **LowerCase**
* `mohmmad -> Mohmmad `
* **Misspelling**
* `Eas -> East`
* **Marathi names**
```

```
* `rasta` -> Road`  
* **UpperCase Words**  
* `chembur` -> Chembur`
```

## 3. Data Overview

### File sizes:

- sample3.osm: 74.1 MB
- nodes\_csv: 16.4 MB
- nodes\_tags.csv: 204 KB
- ways\_csv: 1.67 MB
- ways\_nodes.csv: 5.63 MB
- ways\_tags.csv: 1.05 MB
- mumbai.db: 28.7 MB

### ###Number of nodes:

```
sqlite> SELECT COUNT(*) FROM nodes
```

### Output:

```
204412
```

### Number of ways:

```
sqlite> SELECT COUNT(*) FROM ways
```

### Output:

```
28327
```

### ###Top contributing users:

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

### Output:

```
('PlaneMad', 7735),  
('anushap', 7418),  
('parambyte', 7360),  
('Ashok09', 6523),  
('premkumar', 6314),  
('Srikanth07', 6018),  
('Narsimulu', 5947),  
('sampath reddy', 5555),  
('Naresh08', 5513),  
('pvprasad', 5068)
```

### ###Number of users contributing only once:

```
sqlite> SELECT COUNT(*)  
FROM  
(SELECT e.user, COUNT(*) as num  
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
```

```
GROUP BY e.user
HAVING num=1) u;
```

Output:

```
44
```

## 4. Additional Data Exploration

---

###Common ammenities:

```
sqlite> SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

Output:

```
('bank', 41),
('restaurant', 41),
('place_of_worship', 22),
('atm', 18),
('cafe', 18),
('school', 17),
('toilets', 16),
('fast_food', 15),
('hospital', 14),
('fuel', 10)
```

###Biggest religion:

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='place_of_worship') i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='religion'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 1;
```

Output:

```
Hindu :8
```

###Popular cuisines

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC;
```

Output:

```
('indian', 5),
('regional', 4),
('burger', 2),
('South_Indian', 1),
('Vegetarian_Restaurant', 1),
('chinese', 1),
```

```
('italian', 1),  
('pizza', 1),  
('seafood', 1)
```

## 5. Conclusion

---

The OpenStreetMap data of Mumbai is of fairly reasonable quality but the typo errors caused by the human inputs are significant. We have cleaned a significant amount of the data which is required for this project. But, there are lots of improvement needed in the dataset. The dataset contains very less amount of additional information such as amenities, tourist attractions, popular places and other useful interest. The dataset contains very old information which is now incomparable to that of Google Maps or Bing Maps.

So, I think there are several opportunities for cleaning and validation of the data in the future.

### Additional Suggestion and Ideas

#### Control typo errors

- We can build parser which parse every word input by the users.
- We can make some rules or patterns to input data which users follow everytime to input their data. This will also restrict users input in their native language.
- We can develop script or bot to clean the data regularly or certain period.

#### Benefits of implementing suggestions

- If above mentioned suggestions are followed then there will be little or no irregularities in the data.
- It will be very easy to further process any data and directly feed into machines.

#### More information

- The tourists or even the city people search map to see the basic amenities provided in the city or what are the popular places and attractions in the city or near outside the city. So, the users must be motivated to also provide these informations in the map.
- If we can provide these informations then there are more chances to increase views on the map because many people directly enter the famous name on the map.

## Files

---

- README.md : this file
- sample\_mumbai.osm : sample data of the OSM file
- audit\_street.py : audit street, city and update their names
- audit\_bank.py : audit bank and update their names
- data.py : build CSV files from OSM and also parse, clean and shape data
- tags.py : find unique tags in the data
- query.py : Create database and do different queries about the database using SQL
- report.pdf : pdf of this document
- sample.py : extract sample data from the OSM file
- tags.py : count multiple patterns in the tags

