

OpenStreetMap Data Case Study - Project 3

Author: Jaks Wildman

Map Area: I selected Columbus, Ohio, USA as it is the city I've spent the most time in.

- Location: Columbus, Ohio
- [OpenStreetMap URL](#)

Data Audit

Unique Tags

The XML file utilizes many tags to structure the data. Using `mapparser.py` I counted the number of each unique tag respectively from `columbusOH.osm`. The code for this is `mapparser.py` taken from the class case study.

- `'node' : 1502751`
- `'member' : 38206,`
- `'nd' : 1837440,`
- `'tag' : 687442,`
- `'bounds' : 1, S`
- `'note' : 526,`
- `'meta' : 91356,`
- `'relation' : 77866`
- `'way' : 177854`
- `'osm' : 1`

Patterns

This set of pattern checks was run on the entire osm file vs the sample that's in the root folder and utilizes regular expressions. `tags.py` contains the necessary code to count these 4 categories.

- `"lower" : 429525` - Tags that only contain lowercase letters and pass the validity checks.
- `"lower_colon" : 244778` - Tags that are otherwise valid, yet have colons in their name.
- `"problemchars" : 0` - Tags with problematic characters such as "=", "+", "&", ",", "?" and more.
- `"other" : 13139` - Any other tags that don't fit in the 3 prior categories

Problems In Data

The most significant issue that had to be addressed was the inconsistency in the street names and their abbreviations.

In order to correct these the following functions were utilized within the `audit.py` file:

- `audit_street_type` : Determines if the street name is within the list of expected names
- `street_name` : Tests whether the 'k' attribute matches the key for street data (`addr:street`)
- `audit` : Returns a dictionary of key, value pairs which meet the criteria in the preceding functions
- `update_name` : Actually does the update on the street name

Data Overview

File Sizes

- `columbusOH.osm` : 324 MB
- `nodes.csv` : 124MB
- `nodes_tags.csv` : 3.37 MB
- `ways.csv` : 10.5 MB
- `ways_nodes.csv` : 43.7 MB
- `ways_tags.csv` : 20.2 MB
- `cbus.db` : 171 MB

Number of Nodes:

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

Output: `1502751`

Number of Ways:

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

Output: 177854

A Count by type of the top 15 Node Tags

```
sqlite> SELECT DISTINCT TYPE, COUNT(ID) as TYPE_COUNT
FROM NODES_TAGS
GROUP BY TYPE
ORDER BY TYPE_COUNT DESC
LIMIT 15 ;
```

Output:

regular	76620
addr	7879
gnis	4409
species	1396
fire_hydrant	572
traffic_signals	404
brand	340
contact	121
payment	79
tower	77
name	50
xmas	46
historic	42
service	39
surveillance	38

Number of Unique users:

```
sqlite> SELECT COUNT(DISTINCT(u.uid))
FROM (SELECT uid FROM nodes
      UNION ALL
      SELECT uid from ways) u;
```

Output: 1151

Top Contributors:

```
sqlite> SELECT USER, COUNT(*) AS EDITS
FROM (SELECT USER FROM NODES
      UNION ALL
      SELECT USER FROM WAYS)
GROUP BY USER
ORDER BY EDITS DESC
LIMIT 10;
```

Output:

woodpeck_fixbot	211799
doktorpixel14	167004
Anonononon	157726
Nimbalo	150409
MerlinPendragon	108529
duck57	88834
AndrewSP37	87877
Vid the Kid	69057
kbzimmer	61976
St-Motel	53533

Popular Restaurants by Cuisine

```
sqlite> SELECT NODES_TAGS.VALUE, COUNT(*) AS NUM
FROM NODES_TAGS
JOIN (SELECT DISTINCT(ID) FROM NODES_TAGS
     WHERE VALUE="restaurant") r on nodes_tags.id=r.id
WHERE NODES_TAGS.KEY = 'cuisine'
```

```
GROUP BY nodes_tags.value
ORDER BY NUM DESC
LIMIT 15;
```

Output:

pizza	33
american	26
chinese	22
mexican	16
sandwich	14
italian	10
asian	8
ice_cream	8
indian	6
greek	5
japanese	5
sushi	5
burger	4
chicken;american	3
barbecue	2

Greatest number of Worship Centers by Religion:

```
sqlite> SELECT NODES_TAGS.VALUE, COUNT(*) AS NUM
FROM NODES_TAGS
JOIN (SELECT DISTINCT(ID) FROM NODES_TAGS
      WHERE VALUE = "place_of_worship") a
ON NODES_TAGS.ID = A.ID
WHERE NODES_TAGS.KEY = "religion"
GROUP BY NODES_TAGS.VALUE
ORDER BY NUM DESC
LIMIT 3;
```

Output:

christian	536
muslim	2
jewish	1

Conclusion

Considering the amount of data, and how it is all manually added it is surprisingly clean. However that doesn't mean it IS clean. Each layer reveals more issues that we would need to be addressed such as returning lists of counties. Many times there are more than one in the result. Due to the size of the data, and the non-standard tags it can be very easy to have duplication based on different colloquial names.

Suggestion: Set a reasonable standard on tags for usage so that things like restaurant:cuisine = american and chicken;american doesn't happen. I'm not quite sure what "chicken;america" means.