OpenStreetMap Data Project

Map Area

Baltimore, Md., USA

[https://www.openstreetmap.org/relation/133345#map=12/39.2847/-76.6205 (https://www.openstreetmap.org/relation/133345#map=12/39.2847/-76.6205)](https://www.openstreetmap.org/relation/133345#map=12/39.2847/-76.6205)

First that I might regale you with the words of our official municipal anthem: Baltimore, Our Baltimore...

Baltimore, where Carroll flourished and the fame of Calvert grew! Here the old defenders conquered as their valiant swords they drew. Here the starry banner glistened in the sunshine of the sea, In that dawn of golden vision that awoke the song of Key: Here are hearts that beat forever for the city we adore; Here the love of sisters, brothers: Baltimore, our Baltimore!

Problems Encountered

Having pre-selected Baltimore as the project area, I attempted to download an OSM (XML) dataset of the metro area using mapzen.com. I was deterred by the fact that Mapzen only offers a dataset for the DC/Baltimore metro area, which at 205 MB was too sprawling for this project's purposes. I then resorted to the utilizing the OSM Overpass API to define and download a custom area roughly corresponding to the city boundry, which, after multiple tries, I narrowed down to 39.32 to 39.2724 Latitude and -76.6537 to -76.5659 Longitude, yielding a file of 172.3 MB.

Then, after running data.py on the dataset and visually scanning through 10,000 entries in the resulting ways and nodes_tags.csv's, I noticed a few candidates for cleaning:

```
1.) Under-abbreviated street names ("Baltimore (city), Maryland,Md.,MD,USA
")
2.) Non-capitaized religious denominations ("lutheran")
3.) Alternately spelled words ("neighbourhood")
4.) Inconsistencies in the capitalization of restaurant cuisine designatio
ns ("thai, Afghan")
5.) Numerous inconsistencies in phone number formating including run-on mu
ltiples
```

Under-abbreviated Street Names

Under-abbreviated street names were tackeled in audit.py using a function, fix_problem and a custom mapping to replace the offending string. This step was completed last as it ultimately required altering the regular expression used to parse the string from (r'\S+.?$′, re.IGNORECASE)to(r′$\D + \.?, re.IGNORECASE), to allow for the white space in "Baltimore (city), Maryland,Md.,MD,USA" whcih otherwise passed by unaltered.

In [ ]:

```
def fix_problem(problem, mapping):

    problem_type_re = re.compile(r'\D+\.?$', re.IGNORECASE)
    # Changed from: problem_type_re = re.compile(r'\S+\.?$', re.IGNORECASE)
    #to allow white space in Baltimore (city),Maryland,Md.,MD,USA
    match = problem_type_re.search(problem)
    if match:
        problem_type = match.group()
        if problem_type in mapping.keys():
            mapping.get(problem_type)
            problem = problem.replace(problem_type, mapping.get(problem_type))
    return problem
```

In [ ]:

```
"Baltimore (city), Maryland,Md.,MD,USA" is corrected to "Baltimore, Md."
```

Non-capitaized Religious Denominations

Non-capitalized religious denominations were handled in a similar manner to under-abbreviated street names by the application of a custom mapping in audit.py as well.

In [ ]:

```
"lutheran" is corrected to "Lutheran"
```

Alternate spelling of Neighborhood

For some reason the word neighborhood was consistently misspelled throughout the data as neighbourhood. This is the British spelling and unacceptable in the birth place of the national anthem. This also required a custom mapping in audit.py.

In [ ]:

```
"neighbourhood" is corrected to "neighborhood"
```

Cuisine Spelling Inconsistencies

The values associated with nodes_tags.key='cuisine' were also inconsistent, some proper names capitalized, others not, with one spelling error. These also required a custom mapping.

In [ ]:

```
"ethopian" is corrected to "Ethiopian"
```

Phone# Inconsistencies

Parsing phone numbers required a special function to do so. This function first removes all non-numerical characters, leaving only digits. It then strips any initial long distance prefix and slices the remaining digits into a group of three, another group of three and a group of four, all seperated by white spaces. The addition of an extra phone number in the same v=''/> field posed an additional format problem but this is solved by recursively calling the function from within itself. Nifty.

In [ ]:

```
def format_phone(phonenumber):
    # remove non-digit characters
    phonenumber = ''.join(ele for ele in phonenumber if ele.isdigit())

    #remove leading 1-
    if phonenumber.startswith('1'):
            phonenumber = phonenumber[1:]
    elif phonenumber.startswith('01'):
        phonenumber = phonenumber[:02]

    # One erroneous # of 11 digit length, taking first 10
    if len(phonenumber) == 11:
        phonenumber = format_phone(phonenumber[:10])

    #Recursion!!! To deal with two listed phone #'s
    if len(phonenumber) == 10:
        phonenumber = ('{0}-{1}-{2}'.format(phonenumber[:3], phonenumber[3:6], p
honenumber[6:]))
    elif len(phonenumber) == 21:
        phonenumber = format_phone(phonenumber[:10]) + " " + format_phone(phonen
umber[11:21])
```

In [ ]:

```
"+1 410 385 5848;+1 443 573 2809" is corrected to "410-385-5848 443-573-2809"
```

Overview of the Data

Size of the File

In [ ]:

```
interpreter.osm ......... 172.3 MB
mydb.db .......... 3 MB
nodes.csv ............. 59.3 MB
nodes_tags.csv ........ 2.8 MB
ways.csv .............. 6.2 MB
ways_tags.csv ......... 17.7 MB
ways_nodes.cv ......... 21 MB
```

Number of Nodes

In [ ]:

```
SELECT COUNT(*) FROM nodes_tags
```

In [ ]:

```
84806
```

Number of Ways

In [ ]:

```
SELECT COUNT(*) FROM ways_tags
```

In [ ]:

```
555067
```

Number of Unique Users

In [ ]:

```
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
```

In [ ]:

```
266
```

Top 10 Contributing Users

In [ ]:

```
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;
```

In [ ]:

```
mpetroff-imports, 660571
ElliottPlack, 36575
asciiphil, 14270
mdroads, 13418
westendguy, 3802
woodpeck_fixbot, 1682
RoadGeek_MD99, 1546
mapmaker1559, 1415
BmoreGIS, 1284
woodpeck_repair, 660
```

Number of users appearing only once

In [ ]:

```
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;
```

In [ ]:

```
77
```

Additional Statistics

Pizza is the most popular 'cuisine'...

```
In [ ]:
```

```
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
```

```
In [ ]:
```

```
pizza, 14
American, 13
Italian, 10
Thai, 7
regional, 7
Chinese, 6
Mexican, 6
steak_house, 5
sushi, 5
burger, 4
Korean, 3
sandwich, 3
seafood, 3
Asian, 2
French, 2
Indian;Nepali, 2
Vietnamese, 2
ice_cream, 2
kebab, 2
Afghan, 1
Argentinian, 1
Cuban, 1
Ethiopian, 1
Greek;regional, 1
Indian, 1
Korean;Japanese;Chinese, 1
Latin, 1
Pizza_Italian, 1
Spanish, 1
Sushi_Japanese, 1
bagels, 1
beer,_Belgian, 1
breakfast, 1
breakfast;chicken, 1
churrascaria, 1
contemporary_comfort_food, 1
pho, 1
pizza,_subs, 1
sausages,_lunch, 1
```

Where all the speed cameras are located...

In [ ]:

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='speed_camera') i
    ON nodes_tags.id=i.id
GROUP BY nodes_tags.value
ORDER BY num DESC
```

In [ ]:

```
Speed camera measuring traffic in eastbound direction at intersection of ORLEANS
ST & N LINWOOD AVE, 1
Speed camera measuring traffic in northbound direction at intersection of HARFOR
D RD & E NORTH AVE, 1
Speed camera measuring traffic in northbound direction at intersection of RUSSEL
L ST & W HAMBURG ST, 1
Speed camera measuring traffic in northbound direction at intersection of S MART
IN LUTHER KING JR BLVD & WASHINGTON BLVD, 1
Speed camera measuring traffic in northbound direction at intersection of W MOUN
T ROYAL AVE & W NORTH AVE, 1
Speed camera measuring traffic in southbound direction at intersection of LIGHT
ST & E PRATT ST, 1
Speed camera measuring traffic in southbound direction at intersection of MOUNT
ROYAL TER & W NORTH AVE, 1
Speed camera measuring traffic in southbound direction at intersection of N MONR
OE ST & W LAFAYETTE AVE, 1
Speed camera measuring traffic in southbound direction at intersection of RUSSEL
L ST & BAYARD ST, 1
Speed camera measuring traffic in southbound direction at intersection of RUSSEL
L ST & W HAMBURG ST, 1
Speed camera measuring traffic in westbound direction at intersection of E LOMBA
RD ST & S GAY ST, 1
Speed camera measuring traffic in westbound direction at intersection of E MADIS
ON ST & N CAROLINE ST, 1
Speed camera measuring traffic in westbound direction at intersection of HILLEN
ST & FORREST ST, 1
Speed camera measuring traffic in westbound direction at intersection of W NORTH
AVE & N HOWARD ST, 1
```

A $4 cup of coffee...

In [ ]:

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

In [ ]:

```
$1.50, 1, $1.75, 1, $1.85, 1, $2, 1, $4.00, 1
```

Available leisure activities...

In [ ]:

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

In [ ]:

```
park, 14
playground, 5
sports_centre, 5
fitness_centre, 2
slipway, 2
bowling_alley, 1
dance, 1
garden, 1
pitch, 1
```

A hump, a bump, no those aren't leisure activities: nodes_tags.key='traffic_calming'.

In [ ]:

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

In [ ]:

```
hump, 39, bump, 9, table, 2, choker, 1
```

Horses are more acceptable...

In [ ]:

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
      UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='horse'
GROUP BY tags.value
ORDER BY count DESC;
```

In [ ]:

```
no, 55, yes, 20, private, 2
```

than cryptocurrencies.

In [ ]:

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
          UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='cryptocurrencies'
GROUP BY tags.value
ORDER BY count DESC
```

In [ ]:

```
no, 16
```

Conclusion

In conclusion, I would state that the OpenStreetMap of Baltimore is hindered by the fact that it is an aggregate of the D.C. and Baltimore metro areas.

While from a global perspective, these municipalities may exist in proximity, locally, this is simply too crude an approximation to provide any real insight into either with each featuring their own distinct resources and their own unique challenges.

What is needed is a more locally sourced OSM of the populance, their socio-economic potentialities and predicament.

Although this would require further data mining as well as the expansion of the concept of a map, it would be easily implemented by the creation and utilization of an app.

Maps are just scale models of territory carved out by usage (highways folowing old Native-American trails), and ultimately, made permanent by paving, naming, etc.

The proposed app would collect voluntary anonymized automatic self-reporting of geo-location data from cellular phones to sketch out a 'ghost map' of generated user activity. This data would suffice to serve as a more authentic and complete OSM of the future(primitive).

Although implementation would of course require app creation and user adoption, I believe the robustness of the data generated and its use value would ultimately serve as an incentive to overcome the challenges of its creation.

The use of gameification and rewards to encourage for user adoption and self-reporting could further speed the project's realization and implementation.