OpenStreetMap Data Case Study

Área

Rio de Janeiro, Brasil

- https://mapzen.com/data/metro-extracts/metro/rio-de-janeiro_brazil/ (https://mapzen.com/data/metro-extracts/metro/rio-de-janeiro_brazil/)

Preparação para rodar o caso em python

In [1]:

```
from xml.etree import cElementTree as ET
import sqlite3 as lite
import os
import pandas as pd
import P3
```

Problemas

Vamos falar dos problemas.

Problema 1: A definição das ruas, avenidas, algumas vezes estão faltando e outras não estão padronizadas.

Abaixo os exemplos de ruas com nomes errados.

In [2]:

```
expected_names = ['Rua', 'Avenida', 'Praia', 'Travessa', 'Praça', 'Estrada', 'Ladeira',
 'Boulevard', 'Beco', 'Via', 'Largo',
                'Campo', 'Mirante', 'Acesso', 'Alameda', 'Rodovia', 'Parque', 'Auto',
'RJ-125']
# Convertendo todos os dados para unicode, assim pode ser comparado
expected_names = [name.decode('utf-8') for name in expected_names]

street_types_unexpected = P3.audit_1('rio-de-janeiro_brazil.osm', expected_names)

P3.print_unexpected(street_types_unexpected)
```

```
Servidão
--- Servidão de Passagem 1
--- Servidão de Passagem 1
15
--- 15 de Novembro
Pça.
--- Pça. da Bandeira
Praca
--- Praca Marechal Floriano
Bernadino
--- Bernadino dos Santos
```

A maioria é necessário incluir na frente a palavra "Rua". As demais, "R." e "rua" viram "Rua, "Av" vira "Avenida", "trav" vira "Travessa", "Estr." vira "Estrada", "Pça." e "Praca" viram "Praça".

Problema 2: Lugares estranhos e fora do padrão

Segue abaixo todos os lugares na base de dados.

In [3]:

```
store_types = P3.audit_2('rio-de-janeiro_brazil.osm')

print("Uma pequena amostra dos dados:")
counter = 0
for key in sorted(store_types.iterkeys()):
    counter += 1
    print "%s: %s" % (key, store_types[key])
    if counter == 6:
        break
```

```
Uma pequena amostra dos dados:
Academia: 1
Cassino: 1
Clínica: 1
Clínica Odontológica: 1
Curso de Idiomas: 1
Empada Brasil: 1
```

Podemos ver que 'Academia' pode virar 'gym'. Mas muitos são estranhos e podem ser removidos, como 'yes', 'f', 'other'. Sendo assim vamos remover todas as linhas com somente uma entrada, assim podemos retirar as linhas estranhas. No caso de não ser uma linha estranha ela é no mínimo irrelevante por só conter uma ocorrência.

Problema 3: Formato do encoding

Durante a programação houveram vários problemas de conversão unicode, encode, 'UTF-8' e ascii. Eles foram resolvidos no código, mas não foi algo simples.

Preparação da base de dados

Vamos preparar o SQL para fazer análise dos dados.

In [4]:

```
# Criando o arquivo da base de dados
if os.path.isfile('rio-de-janeiro_brazil.db'):
    os.remove('rio-de-janeiro_brazil.db')
con = lite.connect('rio-de-janeiro_brazil.db')

# Criando as tabelas na base de dados
P3.create_tables_sql(con)
con.commit()
con.close()
```

In [5]:

```
# Inserindo os dados na base de dados
con = lite.connect('rio-de-janeiro_brazil.db')
P3.create_data_sql(con,'rio-de-janeiro_brazil.osm')
con.commit()
con.close()
```

Agora a base de dados está feito, precisamos consertar os problemas apresentados.

Problema 1: A definição das ruas, avenidas, algumas vezes estão faltando e outras não estão padronizadas.

Vamos resolver os nomes de ruas errados.

In [6]:

```
con = lite.connect('rio-de-janeiro_brazil.db')

P3.fix_problem1(con)

con.commit()
con.close()
```

Problema 2: A definição das ruas, avenidas, algumas vezes estão faltando e outras não estão padronizadas.

Vamos remover os lugares estranhos e consertar a 'Academia'

In [7]:

```
con = lite.connect('rio-de-janeiro_brazil.db')
cur = con.cursor()
# Transforma 'Academia' em 'gym'
cur.execute('''UPDATE nodes_tags SET value = 'gym' WHERE value = 'Academia' and key =
 'amenity' ''')
# Remove casos com somente uma ocorrência
cur.execute('''DELETE FROM nodes_tags WHERE key = 'amenity' and
            value in (SELECT value FROM nodes_tags WHERE key = 'amenity' GROUP BY value
 HAVING count(*) = 1)''')
con.commit()
con.close()
```

Criando csv

Em um momento da análise é pedido a criação de CSVs, no caso vamos criá-los usando pandas.

In [8]:

```
def create_csv_from_db(table_name):
    '''Pega a tabela na base de dados e converte para csv'''
    con = lite.connect('rio-de-janeiro_brazil.db')
    dataframe = pd.read_sql('''SELECT * FROM ''' + table_name, con)
    con.close()
    dataframe.to_csv(table_name + '.csv', encoding = 'utf-8')

create_csv_from_db('nodes')
create_csv_from_db('ways')
create_csv_from_db('nodes_tags')
create_csv_from_db('ways_tags')
```

Resultados

Vamos agora apresentar estatísticas sobre os dados.

In [9]:

```
def print_file_size(extension):
    '''Verifica todos os arquivos na pasta com a extensão e imprime nome e tamanho em M
B'''
    for file_name in os.listdir('./'):
        if file_name.endswith(extension):
            print(file_name + ": " + "{:.2f}".format(os.path.getsize(file_name)/1024./1
024) + "MB")

print_file_size('.db')
print_file_size('.osm')
print_file_size('.csv')
```

```
rio-de-janeiro_brazil.db: 88.95MB
example.osm: 6.04MB
rio-de-janeiro_brazil.osm: 329.25MB
nodes.csv: 91.59MB
nodes_tags.csv: 5.08MB
ways.csv: 6.42MB
ways_tags.csv: 15.03MB
```

Podemos notar que o arquivo '.db' é menor do que o '.csv' só da tabela nodes.

Verificamos agora o número de usuários.

In [10]:

```
con = lite.connect('rio-de-janeiro_brazil.db')
cur = con.cursor()
cur.execute('''SELECT uid, count(*) as num FROM nodes GROUP BY uid ORDER BY num desc LI
MIT 10''')
users_modify = cur.fetchall()
print('Os usuários com mais modificações foram:')
for user in users_modify:
    print('Id {0} com {1} modificações.'.format(user[0], user[1]))
cur.execute('''SELECT count(*) FROM (SELECT uid, count(*) as num FROM nodes GROUP BY ui
d)''')
users = cur.fetchall()
print('O total de usuários foi {}.'.format(users[0][0]))
con.close()
```

```
Os usuários com mais modificações foram:
Id 893594 com 364055 modificações.
Id 502691 com 162834 modificações.
Id 4008694 com 155256 modificações.
Id 139043 com 143376 modificações.
Id 289524 com 84084 modificações.
Id 481662 com 65527 modificações.
Id 12293 com 54196 modificações.
Id 82797 com 45644 modificações.
Id 69210 com 32644 modificações.
Id 2783588 com 32473 modificações.
O total de usuários foi 1204.
```

Vamos analisar os nodes e ways.

In [11]:

```
con = lite.connect('rio-de-janeiro_brazil.db')
cur = con.cursor()
cur.execute('''SELECT count(*) FROM (SELECT count(*) FROM nodes GROUP BY id)''')
nodes = cur.fetchall()
print('O total de nodes foi {}.'.format(nodes[0][0]))
cur.execute('''SELECT count(*) FROM (SELECT count(*) FROM ways GROUP BY id)''')
ways = cur.fetchall()
print('O total de ways foi {}.'.format(ways[0][0]))
con.close()
```

```
O total de nodes foi 1576848.
O total de ways foi 190756.
```

In [12]:

```
con = lite.connect('rio-de-janeiro_brazil.db')
cur = con.cursor()
cur.execute('''SELECT value, count(*) FROM nodes_tags WHERE key = 'amenity' GROUP BY va
lue''')
stores = cur.fetchall()
print('Uma amostra das lojas e quantidades são:')
counter = 0
for store in stores:
    counter += 1
    print('A loja {0} tem {1} estabelecimentos no mapa.'.format(store[0], store[1]))
    if counter == 10:
        break
con.close()
```

```
Uma amostra das lojas e quantidades são:
A loja arts_centre tem 6 estabelecimentos no mapa.
A loja atm tem 74 estabelecimentos no mapa.
A loja bank tem 488 estabelecimentos no mapa.
A loja bar tem 221 estabelecimentos no mapa.
A loja bbq tem 3 estabelecimentos no mapa.
A loja bench tem 74 estabelecimentos no mapa.
A loja bicycle_parking tem 1469 estabelecimentos no mapa.
A loja bicycle_rental tem 261 estabelecimentos no mapa.
A loja brothel tem 14 estabelecimentos no mapa.
A loja building tem 4 estabelecimentos no mapa.
```

In [13]:

```
# Colocar os 3 usuários que mais modificaram bares
con = lite.connect('rio-de-janeiro_brazil.db')
cur = con.cursor()
cur.execute('''SELECT ways.uid, count(*) as num FROM ways, ways_tags WHERE ways.id = wa
ys_tags.id and
            ways_tags.key = 'amenity' and ways_tags.value = 'bar' GROUP BY ways.uid ORD
ER BY num desc LIMIT 3''')
users = cur.fetchall()
print('Os 3 usuários que mais modificaram bares:')
for user in users:
    print('O usuário {0} modificou {1} bares.'.format(user[0], user[1]))

con.close()
```

```
Os 3 usuários que mais modificaram bares:
O usuário 1746977 modificou 7 bares.
O usuário 1281877 modificou 6 bares.
O usuário 502691 modificou 5 bares.
```

Futuras análises

Pode ser avaliado futuramente informações como a média de nomes que cada usuário teve. Esse tipo de análise não teria muitas dificuldades, pode-se até analisar quais usuários mais tiveram diferentes nomes.

Outra análise seria melhor tratar os tipos de estabelecimentos. O problema é que precisaria de uma análise mais apurada dos estabelecimentos. Por exemplo, para mim 'waste_basket' deve ser o mesmo que 'waste_disposal', mas precisa de uma análise para confirmar se essa informação está certa e como podemos juntá-la.

Uma terceira análise possível seria verificar o CEP dos estabelecimentos usando "Regular Expressions". O problema seria como proceguir para os dados errados. O mesmo poderia ser corrigido com uma pesquisa na internet para cada endereço errado ou simplesmente apagado. O caso perfeito seria a correção dos mesmo, mas isso seria trabalhoso.

Referências

- http://stackoverflow.com/ (http://stackoverflow.com/)
- https://www.openstreetmap.org (https://www.openstreetmap.org)
- https://mapzen.com/data/metro-extracts/ (https://mapzen.com/data/metro-extracts/)
- https://wiki.openstreetmap.org (https://wiki.openstreetmap.org)