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
```

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]

def is_street_name(elem):
    '''Verifica se o elemento é um nome de rua'''
    return (elem.attrib['k'] == 'addr:street')

def first_word(street_name):
    '''Pega a primeira palavra da string'''
    return street_name.split()[0]

def audit(osm_file):
    '''Verifica o início do nome da rua que não era esperado e coloca todas as ocorrênc
ias em um dicionário'''
    street_types_unexpected = {}
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == 'way':
            for tag in elem.iter('tag'):
                if is_street_name(tag):
                    if first_word(tag.attrib['v']) not in expected_names:
                        if first_word(tag.attrib['v']) in street_types_unexpected:

street_types_unexpected[first_word(tag.attrib['v'])].append(tag.attrib['v'])
                        else:
                            street_types_unexpected[first_word(tag.attrib['v'])] =
[tag.attrib['v']]
    return street_types_unexpected

def print_unexpected(street_types_unexpected):
    '''Apresenta as ruas com nome para ser aprimoradas'''
    for key, values in street_types_unexpected.iteritems():
        print(key)
        for value in values:
            print("--- " + value)

street_types_unexpected = audit('rio-de-janeiro_brazil.osm')

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
Professor
--- Professor Fioravanti Di Piero
rua
--- rua G
Estr.
--- Estr. da Paciência
Lourival
--- Lourival Tavares de Paula
Mário
--- Mário Agostinelli
--- Mário Agostinelli
--- Mário Agostinelli
--- Mário Agostinelli
--- Mário Agostinelli
Trav
--- Trav Mario dos Santos
Heráclito
--- Heráclito Graça
Av
--- Av Castelo Branco
--- Av Rotary
--- Av Padre Anchieta
Alfredo
--- Alfredo Ceschiatti
--- Alfredo Ceschiatti
--- Alfredo Ceschiatti
--- Alfredo Ceschiatti
--- Alfredo Ceschiatti
R.
--- R. Ten. Ronald Santoro
--- R. Miguel Gustavo
--- R. Miguel Gustavo
--- R. Miguel Gustavo
--- R. Miguel Gustavo
--- R. Miguel Gustavo
--- R. Miguel Gustavo
--- R. Silva Cardoso
Afredo
--- Afredo Ceschiatti
Marquês
--- Marquês de Paraná
Dias
--- Dias Pereira
Presidente
--- Presidente Tancredo Neves
```

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]:

```
def is_amenity(elem):
    '''Verifica se o elemento é um nome de rua'''
    return (elem.attrib['k'] == 'amenity')

def audit(osm_file):
    '''Verifica o início do nome da rua que não era esperado e coloca todas as ocorrênc
ias em um dicionário'''
    store_types = {}
    for event, elem in ET.iterparse(osm_file, events=("start",)):
        if elem.tag == 'node':
            for tag in elem.iter('tag'):
                if is_amenity(tag):
                    if tag.attrib['v'] in store_types:
                        store_types[tag.attrib['v']] += 1
                    else:
                        store_types[tag.attrib['v']] = 1
    return store_types

store_types = audit('rio-de-janeiro_brazil.osm')

for key in sorted(store_types.iterkeys()):
    print "%s: %s" % (key, store_types[key])
```

```
Academia: 1
Cassino: 1
Clínica: 1
Clínica Odontológica: 1
Curso de Idiomas: 1
Empada Brasil: 1
Estofadora: 1
Lanchonete e Pizzaria: 1
Loteria: 1
Mil sabores: 1
Ponto de Taxis: 1
Produtos Defumados: 1
Reform 455: 1
Rio Decor: 1
UNILA: 1
arts_centre: 6
atm: 74
baby_hatch: 1
bank: 488
bar: 221
bbq: 3
bench: 74
bicycle_parking: 1469
bicycle_rental: 261
bicycle_repair_station: 1
brothel: 14
building: 4
bureau_de_change: 7
bus_station: 155
cafe: 127
car_fixing: 3
car_rental: 9
car_wash: 25
cartorio: 1
casino: 1
charging_station: 1
childcare: 13
ciep 113: 1
cinema: 28
clinic: 36
clock: 21
clothes_fixing: 1
clothes_washing: 1
club: 1
college: 15
community_centre: 8
courthouse: 9
dentist: 21
doctors: 29
dojo: 2
drinking_water: 12
driving_school: 5
eletronics_fix: 1
elevator: 2
embassy: 36
estofador: 1
f: 1
fast_food: 821
ferry_terminal: 12
fire_station: 37
fisioterapeuta: 1
```

```
food_court: 3
fountain: 30
fuel: 445
grave_yard: 2
gym: 17
hospital: 154
ice_cream: 5
keymaker: 5
kindergarten: 133
lavoir: 1
library: 15
locksmith: 2
love_hotel: 21
marketplace: 13
masonic_lodge: 1
medical: 1
music_venue: 1
nightclub: 19
organic;fast_food: 1
other: 1
parking: 222
parking_entrance: 53
pet: 1
pharmacy: 344
place_of_worship: 390
police: 159
post_box: 45
post_office: 66
pub: 390
public_building: 38
recycling: 8
refresher: 1
register_office: 6
rescue_station: 1
restaurant: 1027
school: 1536
seamstress: 3
security_booth: 1
shelter: 15
shower: 1
social_facility: 2
stationery: 1
studio: 11
taxi: 122
tec_common: 6
telephone: 371
theatre: 39
toilets: 158
townhall: 14
university: 24
vending_machine: 7
veterinary: 31
waste_basket: 109
waste_disposal: 4
yes: 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
cur = con.cursor()
cur.execute('''CREATE TABLE nodes (
id INTEGER PRIMARY KEY NOT NULL,
lat REAL,
lon REAL,
user TEXT,
uid INTEGER
);''')
cur.execute('''CREATE TABLE nodes_tags (
id INTEGER,
key TEXT,
value TEXT,
FOREIGN KEY (id) REFERENCES nodes(id)
);''')
cur.execute('''CREATE TABLE ways (
id INTEGER PRIMARY KEY NOT NULL,
user TEXT,
uid INTEGER
);''')
cur.execute('''CREATE TABLE ways_tags (
id INTEGER NOT NULL,
key TEXT NOT NULL,
value TEXT NOT NULL,
FOREIGN KEY (id) REFERENCES ways(id)
);''')
con.commit()
con.close()
```

In [5]:

```
# Inserindo os dados na base de dados
con = lite.connect('rio-de-janeiro_brazil.db')
cur = con.cursor()
for event, elem in ET.iterparse('rio-de-janeiro_brazil.osm', events=("start",)):
    if elem.tag == 'node':
        cur.execute("INSERT INTO nodes VALUES(" + elem.attrib['id'] + "," +
elem.attrib['lat'] + ","
                    + elem.attrib['lon'] + ",'" + elem.attrib['user'] + "'," + elem.att
rib['uid'] + ")")
        for tag in elem.iter('tag'):
            cur.execute("INSERT INTO nodes_tags VALUES(" + elem.attrib['id'] + ",'" + t
ag.attrib['k'] + "','"
                            + tag.attrib['v'].replace("'","") + "')")
    if elem.tag == 'way':
        cur.execute("INSERT INTO ways VALUES(" + elem.attrib['id'] + ",'"
                    + elem.attrib['user'] + "'," + elem.attrib['uid'] + ")")
        for tag in elem.iter('tag'):
            cur.execute("INSERT INTO ways_tags VALUES(" + elem.attrib['id'] + ",'" + ta
g.attrib['k'] + "','"
                    + tag.attrib['v'].replace("'","") + "')")
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]:

```
words_to_correct = ['Servidão', '15', 'Pça.', 'Praca', 'Bernadino', 'Professor', 'rua',
 'Estr.', 'Lourival',
                    'Mário', 'Trav', 'Heráclito', 'Av', 'Alfredo', 'R.', 'Afredo', 'Mar
quês', 'Dias', 'Presidente']
# Convertendo todos os dados para unicode, assim pode ser comparado
words_to_correct = [name.decode('utf-8') for name in words_to_correct]

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

cur.execute("SELECT * FROM ways_tags WHERE key = 'addr:street'")
list_to_correct = cur.fetchall()
for address in list_to_correct:
    if first_word(address[2]) in words_to_correct:
        if (first_word(address[2]) == 'R.'.decode('utf-8') or
            first_word(address[2]) == 'rua'.decode('utf-8')):
            new_part = 'Rua'
            old_part = address[2].split()[1:]
            new_word = new_part + ' ' + ' '.join(old_part)
            cur.execute("UPDATE ways_tags SET value = '" + new_word + "' WHERE id = " +
 str(address[0]) + " and key = 'addr:street'")
        elif first_word(address[2]) == 'Av'.decode('utf-8'):
            new_part = 'Avenida'
            old_part = address[2].split()[1:]
            new_word = new_part + ' ' + ' '.join(old_part)
            cur.execute("UPDATE ways_tags SET value = '" + new_word + "' WHERE id = " +
 str(address[0]) + " and key = 'addr:street'")
        elif first_word(address[2]) == 'trav'.decode('utf-8'):
            new_part = 'Travessa'
            old_part = address[2].split()[1:]
            new_word = new_part + ' ' + ' '.join(old_part)
            cur.execute("UPDATE ways_tags SET value = '" + new_word + "' WHERE id = " +
 str(address[0]) + " and key = 'addr:street'")
        elif first_word(address[2]) == 'Estr.'.decode('utf-8'):
            new_part = 'Estrada'
            old_part = address[2].split()[1:]
            new_word = new_part + ' ' + ' '.join(old_part)
            cur.execute("UPDATE ways_tags SET value = '" + new_word + "' WHERE id = " +
 str(address[0]) + " and key = 'addr:street'")
        elif (first_word(address[2]) == 'Pça.'.decode('utf-8') or
            first_word(address[2]) == 'Praca'.decode('utf-8')):
            new_part = 'Praça'
            old_part = address[2].split()[1:]
            new_word = new_part.decode('utf-8') + ' ' + ' '.join(old_part)
            cur.execute("UPDATE ways_tags SET value = '" + new_word + "' WHERE id = " +
 str(address[0]) + " and key = 'addr:street'")
        else:
            new_part = 'Rua'
            old_part = address[2].split()
            new_word = new_part + ' ' + ' '.join(old_part)
            cur.execute("UPDATE ways_tags SET value = '" + new_word + "' WHERE id = " +
 str(address[0]) + " and key = 'addr:street'")
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
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('Lojas e quantidades são:')
for store in stores:
    print('A loja {0} tem {1} estabelecimentos no mapa.'.format(store[0], store[1]))
con.close()
```

```
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.
A loja bureau_de_change tem 7 estabelecimentos no mapa.
A loja bus_station tem 155 estabelecimentos no mapa.
A loja cafe tem 127 estabelecimentos no mapa.
A loja car_fixing tem 3 estabelecimentos no mapa.
A loja car_rental tem 9 estabelecimentos no mapa.
A loja car_wash tem 25 estabelecimentos no mapa.
A loja childcare tem 13 estabelecimentos no mapa.
A loja cinema tem 28 estabelecimentos no mapa.
A loja clinic tem 36 estabelecimentos no mapa.
A loja clock tem 21 estabelecimentos no mapa.
A loja college tem 15 estabelecimentos no mapa.
A loja community_centre tem 8 estabelecimentos no mapa.
A loja courthouse tem 9 estabelecimentos no mapa.
A loja dentist tem 21 estabelecimentos no mapa.
A loja doctors tem 29 estabelecimentos no mapa.
A loja dojo tem 2 estabelecimentos no mapa.
A loja drinking_water tem 12 estabelecimentos no mapa.
A loja driving_school tem 5 estabelecimentos no mapa.
A loja elevator tem 2 estabelecimentos no mapa.
A loja embassy tem 36 estabelecimentos no mapa.
A loja fast_food tem 821 estabelecimentos no mapa.
A loja ferry_terminal tem 12 estabelecimentos no mapa.
A loja fire_station tem 37 estabelecimentos no mapa.
A loja food_court tem 3 estabelecimentos no mapa.
A loja fountain tem 30 estabelecimentos no mapa.
A loja fuel tem 445 estabelecimentos no mapa.
A loja grave_yard tem 2 estabelecimentos no mapa.
A loja gym tem 18 estabelecimentos no mapa.
A loja hospital tem 154 estabelecimentos no mapa.
A loja ice_cream tem 5 estabelecimentos no mapa.
A loja keymaker tem 5 estabelecimentos no mapa.
A loja kindergarten tem 133 estabelecimentos no mapa.
A loja library tem 15 estabelecimentos no mapa.
A loja locksmith tem 2 estabelecimentos no mapa.
A loja love_hotel tem 21 estabelecimentos no mapa.
A loja marketplace tem 13 estabelecimentos no mapa.
A loja nightclub tem 19 estabelecimentos no mapa.
A loja parking tem 222 estabelecimentos no mapa.
A loja parking_entrance tem 53 estabelecimentos no mapa.
A loja pharmacy tem 344 estabelecimentos no mapa.
A loja place_of_worship tem 390 estabelecimentos no mapa.
A loja police tem 159 estabelecimentos no mapa.
A loja post_box tem 45 estabelecimentos no mapa.
A loja post_office tem 66 estabelecimentos no mapa.
A loja pub tem 390 estabelecimentos no mapa.
A loja public_building tem 38 estabelecimentos no mapa.
A loja recycling tem 8 estabelecimentos no mapa.
A loja register_office tem 6 estabelecimentos no mapa.
A loja restaurant tem 1027 estabelecimentos no mapa.
A loja school tem 1536 estabelecimentos no mapa.
```

```
A loja seamstress tem 3 estabelecimentos no mapa.
A loja shelter tem 15 estabelecimentos no mapa.
A loja social_facility tem 2 estabelecimentos no mapa.
A loja studio tem 11 estabelecimentos no mapa.
A loja taxi tem 122 estabelecimentos no mapa.
A loja tec_common tem 6 estabelecimentos no mapa.
A loja telephone tem 371 estabelecimentos no mapa.
A loja theatre tem 39 estabelecimentos no mapa.
A loja toilets tem 158 estabelecimentos no mapa.
A loja townhall tem 14 estabelecimentos no mapa.
A loja university tem 24 estabelecimentos no mapa.
A loja vending_machine tem 7 estabelecimentos no mapa.
A loja veterinary tem 31 estabelecimentos no mapa.
A loja waste_basket tem 109 estabelecimentos no mapa.
A loja waste_disposal tem 4 estabelecimentos no mapa.
```

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.

In [ ]: