OpenStreetMap Sample Project Data Wrangling with MongoDB

https://www.openstreetmap.org/relation/5750005 (https://www.openstreetmap.org/relation/5750005)
https://mapzen.com/data/metro-extracts/metro/sydney_australia/ (https://mapzen.com/data/metro-extracts/metro/sydney_australia/)

1. 在地图中遇到的问题
2. 数据概述
3. 关于数据集的其他想法

开始导入必要的库指定好文件和需要的正则表达式

In [1]:
```
import xml.etree.ElementTree as ET  # Use cElementTree or lxml if too slow
import pprint
import re
from collections import defaultdict
import codecs
import json

OSM_FILE = "sydney_australia.osm"  # Replace this with your osm file
SAMPLE_FILE = "sample.osm"

lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')
```

生成sample.osm

In [21]:
```python
k = 50 # Parameter: take every k-th top level element

def get_element(osm_file, tags=('node', 'way', 'relation')):
    """Yield element if it is the right type of tag

    Reference:
    http://stackoverflow.com/questions/3095434/inserting-newlines-in-xml-file-generated-via-xml-et
ree-elementtree-in-python
    """
    context = iter(ET.iterparse(osm_file, events=('start', 'end')))
    _, root = next(context)
    for event, elem in context:
        if event == 'end' and elem.tag in tags:
            yield elem
            root.clear()


with open(SAMPLE_FILE, 'wb') as output:
    output.write('<?xml version="1.0" encoding="UTF-8"?>\n')
    output.write('<osm>\n  ')

    # Write every kth top level element
    for i, element in enumerate(get_element(OSM_FILE)):
        if i % k == 0:
            output.write(ET.tostring(element, encoding='utf-8'))

    output.write('</osm>')
```

找出所有的tag并统计各个tag的数量

In [2]:
```python
def count_tags(filename):
    tags={}
    for event,elem in ET.iterparse(filename):
        if tags.has_key(elem.tag):
            tags[elem.tag]+=1
        else:
            tags[elem.tag]=1
    return tags


tags = count_tags(SAMPLE_FILE)
pprint.pprint(tags)
```

```
{'member': 906,
 'nd': 35494,
 'node': 29528,
 'osm': 1,
 'relation': 107,
 'tag': 16957,
 'way': 4054}
```

找出各种tag的类型

In [3]:
```
def key_type(element, keys):
    if element.tag == "tag":
        for tag in element.iter('tag'):
            if lower.search(element.attrib['k']):
                keys['lower']+=1
            elif lower_colon.search(element.attrib['k']):
                keys['lower_colon']+=1
            elif problemchars.search(element.attrib['k']):
                keys['problemchars']+=1
            else:
                keys['other']+=1
    return keys

def process_map(filename):
    keys = {"lower": 0, "lower_colon": 0, "problemchars": 0, "other": 0}
    for _, element in ET.iterparse(filename):
        keys = key_type(element, keys)
    return keys



keys = process_map(SAMPLE_FILE)
pprint.pprint(keys)
```

```
{'lower': 14685, 'lower_colon': 2097, 'other': 175, 'problemchars': 0}
```

找出提供数据的用户id

In [4]:
```
def process_map_users(filename):
    users = set()
    for _, element in ET.iterparse(filename):
        if element.get('uid')<>None:
            users.add(element.get('uid'))
    return users

users = process_map_users(SAMPLE_FILE)
print len(users)
# pprint.pprint(users)
```

```
906
```

修正街道简写

In [6]:
```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)

expected = [
    "Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square",
    "Lane", "Road", "Trail", "Parkway", "Commons"
]

# UPDATE THIS VARIABLE
mapping = {"Rd":"Road"}


def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)


def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")


def audit(osmfile):
    osm_file = open(osmfile, "r")
    street_types = defaultdict(set)
    for event, elem in ET.iterparse(osm_file, events=("start", )):

        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    osm_file.close()
    return street_types


def update_name(name, mapping):

    # YOUR CODE HERE
    l = name.split(' ')
    if mapping.has_key(l[-1]):
        l[-1] = mapping[l[-1]]
        name = " ".join(l)
    return name


st_types = audit(SAMPLE_FILE)
# pprint.pprint(dict(st_types))
for st_type, ways in st_types.iteritems():
    for name in ways:
        better_name = update_name(name, mapping)
```

重新构建数据格式并生成json格式数据

```
In  [7]: CREATED = [ "version", "changeset", "timestamp", "user", "uid"]
```

```
def shape_element(element):
    node = {}
    if element.tag == "node" or element.tag == "way" :
        #build type
        node['type'] = element.tag
        #build created
        created = {}
        pos = []
        address = {}
        node_refs = []
        for elem in element.iter():
            for k in elem.attrib.keys():
                if k in CREATED:
                    created[k] = elem.get(k)
                #buid pos
                elif k == 'lat':
                    pos.append(float(elem.get(k)))
                elif k == 'lon':
                    pos.append(float(elem.get(k)))
                #skip problem
                elif problemchars.search(k):
                    continue
                #build address
                elif k == 'k':
                    l = elem.get(k).split(':')
                    if l[0] == 'addr':
                        if len(l) < 3:
                            address[l[1]] =update_name(elem.get('v'),mapping)
                #build with colon like  or k="xx:xxx" v="xxxx"
                    if len(l) > 1:
                        d = {}
                        len(l)
                        d[l[1]] = elem.get('v')
                        node[l[0]] = d
                    if len(l) == 1:
                        node[l[0]] = elem.get('v')
                else:
                    node[k] = elem.get(k)
                #build node_refs
                if element.tag == 'way' and elem.tag == "nd":
                    node_refs.append(elem.get('ref'))
        node['created'] = created
        pos.reverse()
        node['pos'] = pos
        if address <> {}:
            node['address'] = address
        if node_refs <> []:
            node['node_refs'] = node_refs
        return node
    else:
        return None


def process_map_data(file_in, pretty = False):
    # You do not need to change this file
    file_out = "{0}.json".format(file_in)
    data = []
    with codecs.open(file_out, "w") as fo:
        for _, element in ET.iterparse(file_in):
            el = shape_element(element)
            if el:
                data.append(el)
                if pretty:
                    fo.write(json.dumps(el, indent=2)+"\n")
                else:
                    fo.write(json.dumps(el) + "\n")
    return data

data=process_map_data(SAMPLE_FILE,True)
pprint.pprint(data[0])
pprint.pprint(data[-1])
```

```
{'created': {'changeset': '4228056',
             'timestamp': '2010-03-25T10:25:53Z',
             'uid': '20949',
             'user': 'Ebenezer',
             'version': '3'},
 'id': '324883',
 'pos': [-33.9176762, 151.1888395],
 'type': 'node'}
{'building': 'industrial',
 'created': {'changeset': '54044925',
             'timestamp': '2017-11-24T09:30:04Z',
             'uid': '1723158',
             'user': 'ozhiker2',
             'version': '1'},
 'id': '542553645',
 'node_refs': ['5245410301',
               '5245410300',
               '5245410299',
               '5245410298',
               '5245410297',
               '5245410296',
               '5245410301'],
 'pos': [],
 'ref': '5245410301',
 'type': 'way',
 'v': 'industrial'}
```

将生成的json导入mongodb

```
mongoimport --db OpenStreetMap --collection Sydney --file sample.osm.json
```

问题1. 在地图中遇到的问题

回答:
Sydney的数据还算比较规范或者说数据种类比较少,没有遇到特别难处理的问题,只有部分的Rd简写,也通过完善道路名称解决,同类型的amenity由于是不同的user来收集的还存在表述不一致的问题.在整理记录的时候最大的麻烦是处理数组类型字段的聚合问题其实也就是mongodb用的还不是很熟练,通过查StackOverflow,浏览官方的doc花了1个多小时才解决了预期的问题,其实每一个统计结果都会有大于1种的解决方式,熟练以后,选择自己习惯的思路一样可以解决问题.

问题2. 数据概述

回答:

1. 文件大小:原文件sydney_australia.osm 317MB 样本sample.osm 6.39MB
2. 样本数量:
   db.getCollection('Sydney').find({}).count()
   33582
3. 样本中nodes的数量:
   db.getCollection('Sydney').find({"type":"node"}).count() 29524
4. 样本中ways的数量:
   db.getCollection('Sydney').find({"type":"way"}).count() 4054
5. 提供数据用户数:
   db.getCollection('Sydney').distinct("created.uid").length 903
6. 贡献最大的用户:
   db.getCollection('Sydney').aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},{"$sort":{"count":-1}}, {"$limit":1}]) _id:balcoath count: 2346
7. 只贡献过一次的用户数:
   db.getCollection('Sydney').aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}},{"$group":{"_id":"$count", "num_users":{"$sum":1}}},{"$sort":{"count":1}},{"$limit":1}]) 157
8. 最大纬度:
   db.getCollection('Sydney').aggregate([{$project: {"first_value":{$arrayElemAt: ["$pos", 0]}}},{$group: {_id: 'lat', value: {$max: '$first_value'}}}])
   -33.6370504
9. 最小纬度:
   db.getCollection('Sydney').aggregate([{$project: {"first_value":{$arrayElemAt: ["$pos", 0]}}},{$group: {_id: 'lat', value: {$min: '$first_value'}}}])
   -34.1889184
10. 最大经度:
    db.getCollection('Sydney').aggregate([{$project: {"first_value":{$arrayElemAt: ["$pos", 1]}}},{$group: {_id: 'lon', value: {$max: '$first_value'}}}])
    151.3311205
11. 最小经度:
    db.getCollection('Sydney').aggregate([{$project: {"first_value":{$arrayElemAt: ["$pos", 1]}}},{$group: {_id: 'lon', value: {$min: '$first_value'}}}])
    150.628001
12. 多少种设施:
    db.getCollection('Sydney').aggregate([{"$match":{"amenity":{"$exists":1}}},{"$group":{"_id":"$amenity","count":{"$sum":1}}},{"$sort":{"count":-1}}])
    有50种,其中前十的是(1) parking 77(2) bench 30(3) school 27(4) restaurant 27(5) toilets 21(6) cafe 19(7) drinking_water 14(8) fast_food 13(9) bicycle_parking 13(10) place_of_worship 11

问题3. 关于数据集的其他想法

回答:
选择Sydney作为数据源的主要原因是我明年初打算去Austrila旅游,但是从sample.osm该数据包含的内容还是比较简单的,主要是绘制了地理上的一些信息,对于我即将出发去悉尼来说还需要很多相关的信息,比如说交通具体包括了地铁站,地铁路线,公交车站和公交车路线,有餐馆的信息,但是我需要可以量化顾客的反馈,还有平均的用餐价格,酒店的价格和好评度,有停车位最好还有平均的停车价格,还有各个地区的特点,比如地区收入水平,人口密度,街道的安全指数,交通的拥堵程度,还有地理设施的旅游观光指数,甚至有动态的信息,比如交通拥堵情况,房价情况,自然气候情况等等.

建议1:

与城市的交通部门合作,在地图里添加城市交通信息,实时交通情况.

好处:

对于出行人来说可以可以根据交通情况来选择适合的出行方式和交通路线.

预期的问题

交通信息和路况信息数据量大实时性高,还会受到路修改道的影响,收集信息和维护成本高昂.

建议2:

与第三方的商业数据机构合作,在地图的商业场所增加更多的信息比如营业时间,平均消费程度,客户好评度,客流情况,停车及交通情况.

好处:

对于游客和前往消费的人来说这些信息会对改场所在去之前有预期,在选择的时候减少试错成本.

预期的问题

数据的真实性需要把关,涉及商业利益收集信息的时候要严格第三方数据监管,这样也会产生成本,同时数据透明性和监督也需要具体的落实.

建议3:

增加人口密度分布和房屋平均价格分布信息.

好处:

本地人买房的时候可以获取大方向的建议,外地游客可以根据密度和房屋价格再结合交通信息来选择居住的hotel.

预期的问题

这两个信息具有一定的实时性,统计的方法可以选择回归,但是基础数据的收集和更新的成本也很高,取决收集方式,人口可以通过手机基站来获得,地产价格的话只能通过相关的地产市场的数据分析,这两种收集方式对于开源地图来说难度都很大.

```
In [ ]:
```