郑州市OSM数据整理

本次项目选取了我的家乡，河南省郑州市的地理数据作为分析对象。

https://www.openstreetmap.org/relation/3283765

osm文件数据为范围从北纬34.1061000至35.1435000，东经112.3805000至114.5366000的区域的数据。

文件中的数据存在的问题

问题

- 文件中的数据超过了郑州市的区划范围，包含了周边的一些地区的信息。
- 一些属性可以再划分为二级结构，例如"is_in:country"。
- 时间格式在转换为json格式时，由于编码问题无法直接转换。

解决

数据范围问题

- 思路：从数据文件中，可找到id为“3283765”的relation节点，此节点代表了郑州市的行政区划。此节点的"members"成员包括若干way节点和若干relation节点。其中way节点的members又各自包含若干node节点，relation节点代表了
- 代码：

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 20 16:35:49 2018
```

```
@author: Bill
"""
import xml.etree.cElementTree as ET

data_node=[]
data_way=[]

# 获取郑州市辖区内的所有节点
def getzhengzhoutags(file_in, pretty = False):
    tree=ET.parse(file_in)
    r_zhengzhou = tree.find("relation[@id='3283765']")
    get_subtag(r_zhengzhou,tree)
    for m in r_zhengzhou.findall("member[@type='relation']"):
        r = tree.find("relation[@id='"+m["ref"]+"']")
        get_subtag(r,tree)


# 获取relation节点下的所有way节点，以及这些way下的node节点
def get_subtag(relation,tree):
    for m in relation.findall("member[@type='way']"):
        w = tree.find("way[@id='"+m["ref"]+"']")
        data_way.append(w)
        for n in w.findall("nd"):
            n_tag = tree.find("node[@id='"+n["ref"]+"']")
            data_node.append(n_tag)
```

二级结构问题

- 思路：将带有冒号的属性前后拆分，冒号前作为结构体，冒号后作为结构体的属性。
- 代码：

```
# 将带有冒号的属性处理为二级结构
def get_sub_struct(element):
    tags={}
    for t in element.findall("tag"):
        if ":" in t.attrib['k']:
            strs = t.attrib['k'].split(":",num=1)
            tags[strs[0]][strs[1]] = t.attrib['v']
    return tags
```

时间格式Json编码问题

- 思路：重写json标准库中的JSONEncoder,对时间格式进行特别处理，使其适用于我们的文档中的时间结构。
- 代码：

```
import json

# 对时间格式特别处理的JSONEncoder
class CJsonEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, datetime.datetime):
            return obj.strftime('%Y-%m-%d %H:%M:%S')
        elif isinstance(obj, datetime.date):
            return obj.strftime('%Y-%m-%d')
        else:
            return json.JSONEncoder.default(self, obj)
```

数据探索

概述统计

本次采集的数据文件的大小如下：

- zhengzhoumap.osm 77.5MB
- zhengzhoumap_node.json 76.8MB
- zhengzhoumap_way.json 16.7MB

数据的数量：

<code>> db.node.find().count()
360427

> db.way.find().count()
51818</code>

每个用户分别提交的节点类型的数量

<code>> db.node.aggregate([{group : {_id : "created.user" , num:{sum:1}}},{sort:{"num":-1}}])

> db.way.aggregate([{group : {_id : "created.user" , num:{sum:1}}},{sort:{"num":-1}}])</code>

单向道路的数量

<code>> db.way.find({"tags.oneway":true}).count()
7853</code>

关于数据集的其他想法

额外数据探索

行政区划数量统计

首先，我想先统计一下本次采集到的数据集中，各级行政区划的数量，也就是node节点中，tags中包含place=...的数据，并对其进行分类汇总。其中place的值我选取了village,county,city,town和suburb五种，查询语句如下：

<code>> db.node.aggregate([{ match:{ or:[{"tags.place":"city"},{"tags.place":"suburb"},{"tags.place":"village"},{"tags.place":"county"},{"tags.place":"town"}]}},{ group:{_id:" tags.place", num:{$sum:1}}}])

/* 1 */
{
"_id" : "village",
"num" : 168.0
}

/* 2 */
{
"_id" : "city",
"num" : 21.0
}

/* 3 */
{
"_id" : "county",
"num" : 8.0
}

/* 4 */
{
"_id" : "suburb",
"num" : 15.0
}

/* 5 */
{
"_id" : "town",
"num" : 98.0
}</code>

绘制路径

其次，我想尝试一下，根据道路信息的tags.refs中的节点信息的坐标pos信息，来描绘一条道路的路径。

获取连霍高速，也就是ref为G30的道路节点的refs:

<code>> db.way.find({"tags.ref":"G30"},{"tags.refs":1})
/* 1 */
{
"_id" : ObjectId("5ab85f466d77eb18884daf25"),
"tags" : {
"refs" : [
343505836,
343505838,
343505839,
343505841,
343505842,
343505843,
343505845,
343505846,
343505847,
343505849,
343506500,
343506504,
343506508,
343506512,
343506516,
343506519,
343506944,
343506946,
343506947,
343506948,
343506949,
343506951
]
}
}

/* 2 */
...</code>

根据这些refs，也就是node节点的id，获取node节点的pos信息，并按自西向东，自北向南的顺序排序，也就是经度信息递增，纬度信息递减。

<code>> db.node.find({id:{$in:["343505836",
"343505838",
"343505839",
"343505841",...]}},{"pos":1}).sort({"pos[1]":1,"pos[0]":-1})

```
/* 1 */
{
"_id" : ObjectId("5ab629246d77eb29c8d2a59b"),
"pos" : [
34.8631358,
113.3197149
]
}

/* 2 */
{
"_id" : ObjectId("5ab629246d77eb29c8d2a59c"),
"pos" : [
34.8637993,
113.3216978
]
}

/* 3 */
{
"_id" : ObjectId("5ab629246d77eb29c8d2a59d"),
"pos" : [
34.8644213,
113.3238694
]
}

/* 4 */
{
"_id" : ObjectId("5ab629246d77eb29c8d2a59e"),
"pos" : [
34.8647948,
113.3255161
]
}
...
</code>
```

利用这些数据，结合我们截取的地图覆盖范围，就可以计算出原点和这些node节点所代表的二维坐标系位置，再用线连接起来，就可以得到一条本地图覆盖范围内的连霍高速公路路径图。

想法和建议

1-大多数数据都集中在city区划中

在统计各行政区划的过程中，我发现超过80%的节点信息归属于City类型的行政区划。尤其郑州市市区的节点信息更是多于其他下辖市的信息。这一现象也很好理解，毕竟城市的信息化程度相对较高，人口越密集的行政区划中，各种节点的密度以及人们对数字化地图信息的使用程度都较高。针对这种现象，我的建议是采取措施鼓励信息上传者多上传农村和地级市的节点信息,例如可以为隶属不同行政区划的数据分配不同的贡献度权重，使上传村/地级市级数据的用户得到更多的贡献度。

优点：

按照设想，可以得到更多村/地级市级别的地理信息，而且更新能够较为及时。

缺点：

贡献者需要一定的电脑操作水平，有些过于偏远的地区，此措施可能无效。

2-道路路径绘制不够精确

在绘制道路路径的过程中，发现根据节点绘制的路径并不完全精确，因为有些节点的位置距离道路还有一定距离，但也被包括在了way信息中。表示道路位置最精确的应该是道路上的各种指示牌、摄像头、服务区和收费站等。

优点：

能够得到更精确的道路路径信息。城市中因为摄像头也联网且分布较密集，能够不用花费很大力气就得到较精确的道路路径信息。

缺点：

对于高速公路，各个指示牌位置信息的采集需要花费较大人力。对于一些偏远地区的道路，或者路况较复杂的道路，信息采集和上传都存在很大困难。