

## Contents

一 数据源.....	1
二 分析结果及及解决办法.....	1
1) 分析结果 .....	1
2) 解决办法 .....	2
三 数据抽取转换过程.....	2
四 数据统计结果.....	3
1 数据总数.....	3
2 Node 节点的数量.....	3
3 Way 节点的数量 .....	3
4 Suppermarket 的数量 .....	4
5 Highway 的种类以及个数.....	4
五 分析数据的问题.....	5
六 数据处理建议.....	6
七 预期的益处和风险.....	6

### 一 数据源

选取的是上海地区的 OSM 文件，大小 54M。链接如下：

[https://s3.amazonaws.com/metro-extracts.mapzen.com/shanghai\\_china.osm.bz2](https://s3.amazonaws.com/metro-extracts.mapzen.com/shanghai_china.osm.bz2)

### 二 分析结果及及解决办法

#### 1) 分析结果

对原始的 OSM 数据进行分析后，发现如下一些问题

- 相同属性的取值不统一
  1. 当 tag 为 “way “时，k=oneway 对应的 v 值不统一，根据官方的解释，应该为 yes 或者 no 但在数据集中有的值为-1。需要统一为 yes 或者 no。官方解释链接如下：

<http://wiki.openstreetmap.org/wiki/Way>

2. highway 和 Hwy 不一致，需要转换 Hwy 到 highway。

- 当 tag 为 node 或者 way 时，k=name 对应的 v 是中文名字时，有的 node 节点没有 k=name-en 属性。
- 当 tag 为 node 时，user 的值有些是中文。但绝大部分的 user 取值是英文字母。

## 2) 解决办法

根据以上列出的问题，相应的解决办法如下

- 属性取值不统一的，在抽取数据时，把不规范或者不统一的值修改成规范或者统一值。
- 增加 k=name-en 属性，其 v 的取值采用百度开放的翻译 API 接口翻译成英文。

代码片段如下：

```
def translate_func(text):
    httpClient = None
    myurl = '/api/trans/vip/translate'
    fromLang = 'zh'
    toLang = 'en'
    salt = random.randint(32768, 65536)
    sign = appid + text + str(salt) + secretKey
    ml = md5.new()
    ml.update(sign)
    sign = ml.hexdigest()
    myurl = myurl + '?appid=' + appid + '&q=' + urllib.quote(
        text) + '&from=' + fromLang + '&to=' + toLang + '&salt=' + str(salt) + '&sign=' + sign

    try:
        httpClient = httplib.HTTPConnection('api.fanyi.baidu.com')
        httpClient.request('GET', myurl)
        response = httpClient.getresponse()
        result = eval(response.read())
        return result["trans_result"][0]["dst"]
    except Exception, e:
        print e
    finally:
        if httpClient:
            httpClient.close()
```

- 将 user 的取值为中文的，用汉语拼音方式改写。

代码片段如下：

```
if u'\u4e00' <= item.attrib["user"] <= u'\u9fff':
    node["user"] = lazy_pinyin(item.attrib["user"])
else:
    node["user"] = item.attrib["user"]
```

## 三 数据抽取转换过程

- ## 5 创建 MongoDB 数据库和集合

```
> use shaosm
switched to db shaosm
> show dbs
admin      0.000GB
examples  0.000GB
local      0.000GB
> db
shaosm
> db.createCollection(mapdata)
2017-09-22T21:20:28.988+0800 E QUERY [thread1] ReferenceError: mapdata is not defined :
@(<shell>):1:1
> db.createCollection("mapdata")
{ "ok" : 1 }
```

#### 四 数据统计结果

## 1 数据总数

```
> show dbs
admin      0.0000GB
examples   0.0000GB
local      0.0000GB
shaosm     0.007GB
> use shaosm
switched to db shaosm
> db.mapdata.find().count()
83898
```

中文在 mongodb 里显示正常:

```
db.napdata.find({'name':'东方天都东5'}).to_dict()
{'_id': 'ObjectID: '93651112ce928804dc27da1', 'name': 'Eastern day county east 5', 'node_refs': [ '4936510597', '4936510598', '4936510599', '4936510600', '4936510601', '4936510602', '4936510603', '4936510604', '4936510605', '4936510606', '4936510607', '4936510608', '4936510609', '4936510610', '4936510611', '4936510612', '4936510613', '4936510614', '4936510615', '4936510616', '4936510617', '4936510618', '4936510619', '4936510620', '4936510621', '4936510622', '4936510623', '4936510624', '4936510625', '4936510626', '4936510627', '4936510628', '4936510629', '4936510630', '4936510631', '4936510632', '4936510633', '4936510634', '4936510635', '4936510636', '4936510637', '4936510638', '4936510639', '4936510640', '4936510641', '4936510642', '4936510643', '4936510644', '4936510645', '4936510646', '4936510647', '4936510648', '4936510649', '4936510650', '4936510651', '4936510652', '4936510653', '4936510654', '4936510655', '4936510656', '4936510657', '4936510658', '4936510659', '4936510660', '4936510661', '4936510662', '4936510663', '4936510664', '4936510665', '4936510666', '4936510667', '4936510668', '4936510669', '4936510670', '4936510671', '4936510672', '4936510673', '4936510674', '4936510675', '4936510676', '4936510677', '4936510678', '4936510679', '4936510680', '4936510681', '4936510682', '4936510683', '4936510684', '4936510685', '4936510686', '4936510687', '4936510688', '4936510689', '4936510690', '4936510691', '4936510692', '4936510693', '4936510694', '4936510695', '4936510696', '4936510697', '4936510698', '4936510699', '4936510700', '4936510701', '4936510702', '4936510703', '4936510704', '4936510705', '4936510706', '4936510707', '4936510708', '4936510709', '4936510710', '4936510711', '4936510712', '4936510713', '4936510714', '4936510715', '4936510716', '4936510717', '4936510718', '4936510719', '4936510720', '4936510721', '4936510722', '4936510723', '4936510724', '4936510725', '4936510726', '4936510727', '4936510728', '4936510729', '4936510730', '4936510731', '4936510732', '4936510733', '4936510734', '4936510735', '4936510736', '4936510737', '4936510738', '4936510739', '4936510740', '4936510741', '4936510742', '4936510743', '4936510744', '4936510745', '4936510746', '4936510747', '4936510748', '4936510749', '4936510750', '4936510751', '4936510752', '4936510753', '4936510754', '4936510755', '4936510756', '4936510757', '4936510758', '4936510759', '4936510760', '4936510761', '4936510762', '4936510763', '4936510764', '4936510765', '4936510766', '4936510767', '4936510768', '4936510769', '4936510770', '4936510771', '4936510772', '4936510773', '4936510774', '4936510775', '4936510776', '4936510777', '4936510778', '4936510779', '4936510780', '4936510781', '4936510782', '4936510783', '4936510784', '4936510785', '4936510786', '4936510787', '4936510788', '4936510789', '4936510790', '4936510791', '4936510792', '4936510793', '4936510794', '4936510795', '4936510796', '4936510797', '4936510798', '4936510799', '4936510800', '4936510801', '4936510802', '4936510803', '4936510804', '4936510805', '4936510806', '4936510807', '4936510808', '4936510809', '4936510810', '4936510811', '4936510812', '4936510813', '4936510814', '4936510815', '4936510816', '4936510817', '4936510818', '4936510819', '4936510820', '4936510821', '4936510822', '4936510823', '4936510824', '4936510825', '4936510826', '4936510827', '4936510828', '4936510829', '4936510830', '4936510831', '4936510832', '4936510833', '4936510834', '4936510835', '4936510836', '4936510837', '4936510838', '4936510839', '4936510840', '4936510841', '4936510842', '4936510843', '4936510844', '4936510845', '4936510846', '4936510847', '4936510848', '4936510849', '4936510850', '4936510851', '4936510852', '4936510853', '4936510854', '4936510855', '4936510856', '4936510857', '4936510858', '4936510859', '4936510860', '4936510861', '4936510862', '4936510863', '4936510864', '4936510865', '4936510866', '4936510867', '4936510868', '4936510869', '4936510870', '4936510871', '4936510872', '4936510873', '4936510874', '4936510875', '4936510876', '4936510877', '4936510878', '4936510879', '4936510880', '4936510881', '4936510882', '4936510883', '4936510884', '4936510885', '4936510886', '4936510887', '4936510888', '4936510889', '4936510890', '4936510891', '4936510892', '4936510893', '4936510894', '4936510895', '4936510896', '4936510897', '4936510898', '4936510899', '4936510900', '4936510901', '4936510902', '4936510903', '4936510904', '4936510905', '4936510906', '4936510907', '4936510908', '4936510909', '4936510910', '4936510911', '4936510912', '4936510913', '4936510914', '4936510915', '4936510916', '4936510917', '4936510918', '4936510919', '4936510920', '4936510921', '4936510922', '4936510923', '4936510924', '4936510925', '4936510926', '4936510927', '4936510928', '4936510929', '4936510930', '4936510931', '493651093
```

```
> db.mapdata.find().count()
```

83898

## 2 Node 节点的数量

```
> db.mapdata.find({"type":"node"}).count()
```

74650

### 3 Way 节点的数量

```
> db.mapdata.find({"type":"way"}).count()
```

9248

4 Supermarket 的数量

```
> db.mapdata.find({"shop": "supermarket"}).count()
```

3

5 Highway 的种类以及个数

```
> db.mapdata.aggregate([{$group:{_id: "$highway",result: { $sum: 1 }}}])
```

```
{ "_id" : "services", "result" : 1 }
```

```
{ "_id" : "tertiary_link", "result" : 9 }
```

```
{ "_id" : "platform", "result" : 1 }
```

```
{ "_id" : "pedestrian", "result" : 21 }
```

```
{ "_id" : "track", "result" : 62 }
```

```
{ "_id" : "living_street", "result" : 10 }
```

```
{ "_id" : "steps", "result" : 21 }
```

```
{ "_id" : "path", "result" : 36 }
```

```
{ "_id" : "trunk_link", "result" : 64 }
```

```
{ "_id" : "service", "result" : 584 }
```

```
{ "_id" : "raceway", "result" : 2 }
```

```
{ "_id" : "construction", "result" : 25 }
```

```
{ "_id" : "cycleway", "result" : 33 }
```

```
{ "_id" : "motorway_link", "result" : 195 }
```

```
{ "_id" : "footway", "result" : 206 }
```

```
{ "_id" : "motorway", "result" : 247 }
```

```
{ "_id" : "primary_link", "result" : 75 }
```

```
{ "_id" : "trunk", "result" : 60 }
```

```
{ "_id" : "secondary", "result" : 454 }
```

```
{ "_id" : "secondary_link", "result" : 25 }
```

## 五 分析数据的问题

通过第一次数据分析，发现如下一些问题：

1 当 type 是 way（道路）的时候，相关一些属性在有的节点没有记录，比如是否 ‘railway’ 还是 ‘highway’ 等其他属性，并没有记录。这样就造成后面数据统计的时候数据不准确。比如：

.....

```
"user": "Steven Shen",
```

```
"type": "way",
```

```
"id": "240348083",
```

.....

2 当 type 是 way（道路）的时候，其 node\_refs 的值里有些 node 其实并不存在。这些 node 并没有出现在 type 是 node（节点）。比如：

```
{
```

```
"node_refs": [
```

```
  "3911958650", 这个值在 type 是 node（节点）时并不存在。
```

```
  "3911958859"
```

```
],
```

```
"created": {
```

```
  "changeset": "45942077",
```

```
  "version": "2",
```

```
  "uid": "445671",
```

```
  "timestamp": "2017-02-09T09:37:50Z"
```

```
},
```

```
"user": "flierfy",
```

```
"railway": "rail",
```

```
"type": "way",  
"id": "387966082"  
}
```

## 六 数据处理建议

1 当 **type** 是 **way**（道路）时，有些节点属性缺失的情况，可能需要参考其他一些道路数据集，尽量将这些属性补全，如果没有，考虑将这些节点删除掉，因为这些节点对于后面的一些统计分析没有啥价值。

2 当 **type** 是 **way**（道路）时，对于 **node\_refs** 的值里有些 **node** 其实并不存在的情况，同样需要参考其他一些数据集，或者放大采样范围，或许能将这一部分数据补齐。否则建议将这些不存在的 **node** 从 **node\_refs** 里删除掉。

## 七 预期的益处和风险

益处：

补足缺失的属性后，当统计所有 **type** 是 **way**（道路）时，会得到更加准确的结果。比如：之前统计的“**raceway**”的数量为 2，（{**"\_id"**:**"raceway"**,**"result"**:2}）。对 **node** 的真实性实施改进后，**way** 的 **node\_refs** 值会更准确可靠，我个人觉得 **node\_refs** 可以用于导航路线的生成，那么改进后生成的导航路线会更加准确和真实。

风险：

处理分析数据的难度会增大，增加更多的人力和时间成本，体现在如下几点：

- 1 需要选取有效的额外的数据集，这个需要花费大量的时间去搜索和比对，做前期数据的筛选校验。
- 2 如果数据集的存储格式不一致，数据分析处理过程不同，导致程序版本过多，后期迭代开发不方便。
- 3 整体数据分析过程复杂，很难做成统一的数据处理接口，