OpenStreetMap Project Data Wrangling with MangoDB

Tuyamei

Map Area: Melbourne，Australia

https://mapzen.com/data/metro-extracts/metro/melbourne_australia/

the reason I chose Melbourne to analyze is that my boyfriend will go to australia for further study, so I want to explore some information about this place.

1. Problems Encountered in the Map（audit.py）

After initially downloading a small sample size of the Charlotte area and running it against a provisional data.py file, I noticed three main problems with the data, which I will discuss in the following order:

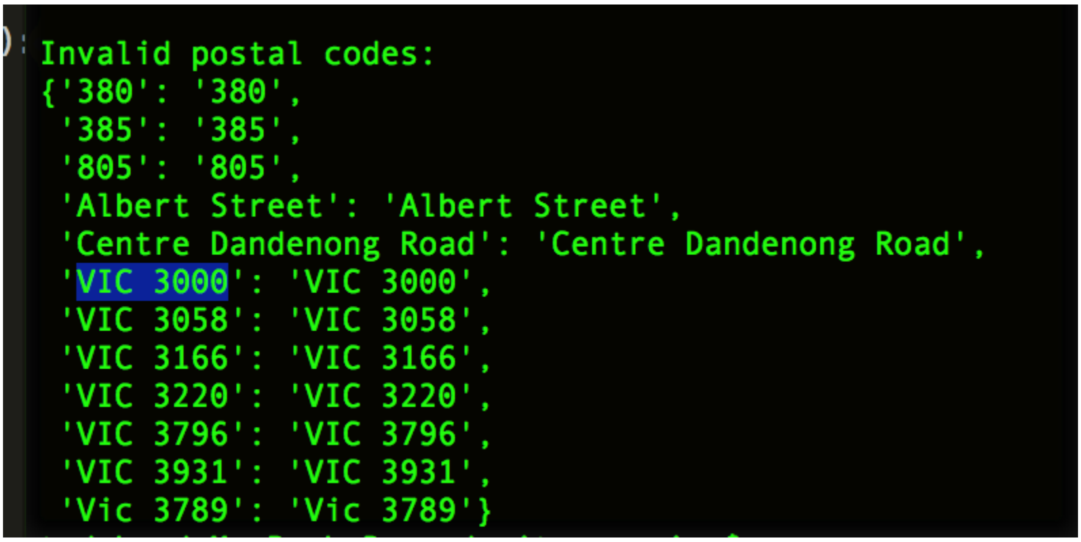
- Over abbreviated street names (“Baxter-Tooradin rd”)
- error spelling ("Maroondah Higway"should be "Highway")
- inconsistent postal code

Over-abbreviated Street Names

Once the data was imported to MongoDB, some basic querying revealed street name abbreviations and postal code inconsistencies. I updated all substrings in problematic address strings, such that “Baxter-Tooradin rd” becomes “Baxter-Tooradin road”.

inconsistent postal code

right code format is the 4-digit number, while using the zipcodes.py, I find out that there are some other invalid format, such as 3-digit(380,385), or other wrong names



```
Invalid postal codes:
{'380': '380',
 '385': '385',
 '805': '805',
 'Albert Street': 'Albert Street',
 'Centre Dandenong Road': 'Centre Dandenong Road',
 'VIC 3000': 'VIC 3000',
 'VIC 3058': 'VIC 3058',
 'VIC 3166': 'VIC 3166',
 'VIC 3220': 'VIC 3220',
 'VIC 3796': 'VIC 3796',
 'VIC 3931': 'VIC 3931',
 'Vic 3789': 'Vic 3789'}
```

2.Data Overview

1.mapparser.py was used to count occurrences of each tag, with a result:

```
{'bounds': 1,
 'member': 103615,
 'nd': 4581896,
 'node': 3898719,
 'osm': 1,
 'relation': 4729,
 'tag': 2314651,
 'way': 535477}
```

2.Before processing the data and add it into your database, I use *tags.py* check the "k" value for each "<tag>" and see if there are any potential problems.

"lower", for tags that contain only lowercase letters and are valid,

"lower_colon", for otherwise valid tags with a colon in their names,

"problemchars", for tags with problematic characters, and

"other", for other tags that do not fall into the other three categories.

```
{'lower': 1722795, 'lower_colon': 589456, 'other': 2400, 'problemchars': 0}
```

3. *users.py* to calculate number of contributing users :

2604

5. *audit.py* to clean the problem of streetname

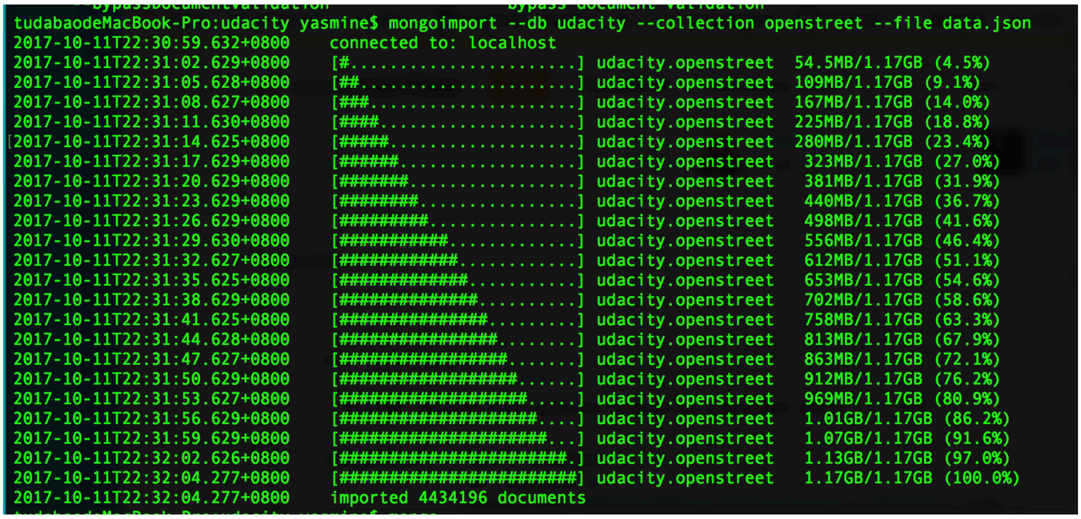
6. use *data.py* to transform the data format from osm to json in order to put data into

7. when import json file into mongodb, I met a lot of problems:

reference:http://www.runoob.com/mongodb/mongodb-dropdatabase.html

http://www.cnblogs.com/now-future/p/6507249.html

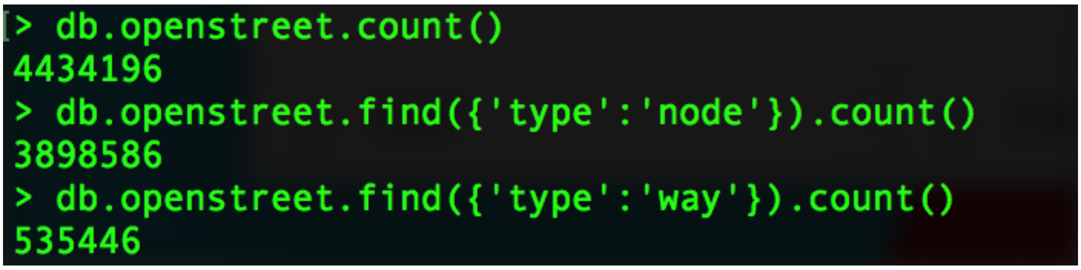
https://stackoverflow.com/questions/30953611/mongodb-error-validating-settings-only-one-positional-argument-is-allowed



```
      --bypassDocumentValidation                  bypass document validation
tudabaodeMacBook-Pro:udacity yasmine$ mongoimport --db udacity --collection openstreet --file data.json
2017-10-11T22:30:59.632+0800    connected to: localhost
2017-10-11T22:31:02.629+0800    [#.......................] udacity.openstreet  54.5MB/1.17GB (4.5%)
2017-10-11T22:31:05.628+0800    [##......................] udacity.openstreet  109MB/1.17GB (9.1%)
2017-10-11T22:31:08.627+0800    [###.....................] udacity.openstreet  167MB/1.17GB (14.0%)
2017-10-11T22:31:11.630+0800    [####....................] udacity.openstreet  225MB/1.17GB (18.8%)
[2017-10-11T22:31:14.625+0800    [#####...................] udacity.openstreet  280MB/1.17GB (23.4%)
2017-10-11T22:31:17.629+0800    [######..................] udacity.openstreet   323MB/1.17GB (27.0%)
2017-10-11T22:31:20.629+0800    [#######.................] udacity.openstreet   381MB/1.17GB (31.9%)
2017-10-11T22:31:23.629+0800    [########................] udacity.openstreet   440MB/1.17GB (36.7%)
2017-10-11T22:31:26.629+0800    [#########...............] udacity.openstreet   498MB/1.17GB (41.6%)
2017-10-11T22:31:29.630+0800    [###########.............] udacity.openstreet   556MB/1.17GB (46.4%)
2017-10-11T22:31:32.627+0800    [############............] udacity.openstreet   612MB/1.17GB (51.1%)
2017-10-11T22:31:35.625+0800    [#############...........] udacity.openstreet   653MB/1.17GB (54.6%)
2017-10-11T22:31:38.629+0800    [##############..........] udacity.openstreet   702MB/1.17GB (58.6%)
2017-10-11T22:31:41.625+0800    [###############.........] udacity.openstreet   758MB/1.17GB (63.3%)
2017-10-11T22:31:44.628+0800    [################........] udacity.openstreet   813MB/1.17GB (67.9%)
2017-10-11T22:31:47.627+0800    [#################.......] udacity.openstreet   863MB/1.17GB (72.1%)
2017-10-11T22:31:50.629+0800    [##################......] udacity.openstreet   912MB/1.17GB (76.2%)
2017-10-11T22:31:53.627+0800    [###################.....] udacity.openstreet   969MB/1.17GB (80.9%)
2017-10-11T22:31:56.629+0800    [####################....] udacity.openstreet   1.01GB/1.17GB (86.2%)
2017-10-11T22:31:59.629+0800    [#####################...] udacity.openstreet   1.07GB/1.17GB (91.6%)
2017-10-11T22:32:02.626+0800    [#######################.] udacity.openstreet   1.13GB/1.17GB (97.0%)
2017-10-11T22:32:04.277+0800    [########################] udacity.openstreet   1.17GB/1.17GB (100.0%)
2017-10-11T22:32:04.277+0800    imported 4434196 documents
tudabaodeMacBook-Pro:udacity yasmine$ mongo
```

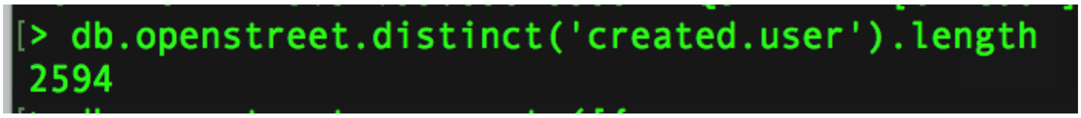
8.use mongodb

>>number of nodes and ways



```
[> db.openstreet.count()
4434196
> db.openstreet.find({'type':'node'}).count()
3898586
> db.openstreet.find({'type':'way'}).count()
535446
```

>>number of users

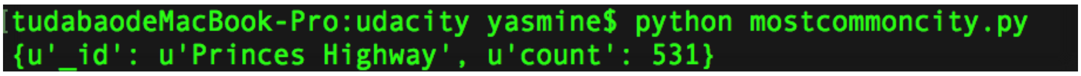


```
[> db.openstreet.distinct('created.user').length
2594
```

3.other ideas

>>the most common city name in our cities collection(mostcommoncity.py)

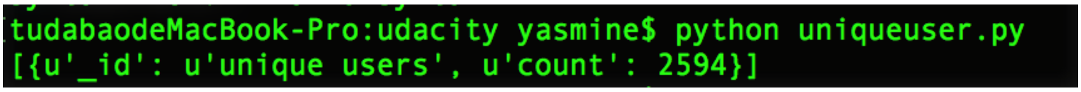
```
pipeline = [
         {"$match":{"name":{"$ne":None}}},
         {"$group":{
               "_id":"$name",
               "count":{"$sum":1}}},
         {"$sort":{"count":-1}},
         {"$limit":1}
         ]
```



```
[tudabaodeMacBook-Pro:udacity yasmine$ python mostcommoncity.py
{u'_id': u'Princes Highway', u'count': 531}
```

>>>number of unique user (uniqueuser.py)

```
pipeline = [
      {
      '$group': {
         "_id": "$created.user"
      }},
      {
      '$group': {
         "_id": "unique users",
         "count": {"$sum": 1},
      }}]
```



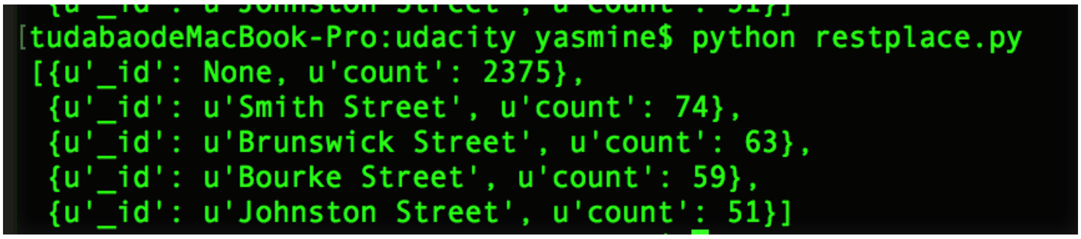
```
tudabaodeMacBook-Pro:udacity yasmine$ python uniqueuser.py
[{u'_id': u'unique users', u'count': 2594}]
```

>>>find the most fun place to have fun (restplace.py)

```
pipeline = [
                {"$match":{"_id":{"$ne":None}}},
                {
                '$match': {

                    '$or': [
                        {"amenity": "restaurant"},
                        {"amenity": "bar"} ,
                        {"amenity": "pub"} ,
                        {"amenity": "cafe"} ,
                    ],
                    #"address.street": {'$exists': true},
                },
            },
            {
                '$group': {
                    "_id": "$address.street",
                    "count": {"$sum": 1},
                },
            },
            {
                '$sort': {
                    "count": -1
                }
            },
            {
                '$limit': 5,
            }
        ]
```

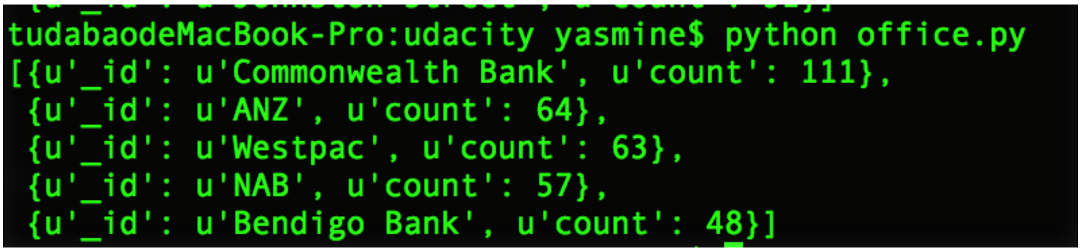
>>response:



```
 {u'_id': u'Johnston Street', u'count': 51}]
[tudabaodeMacBook-Pro:udacity yasmine$ python restplace.py
 [{u'_id': None, u'count': 2375},
  {u'_id': u'Smith Street', u'count': 74},
  {u'_id': u'Brunswick Street', u'count': 63},
  {u'_id': u'Bourke Street', u'count': 59},
  {u'_id': u'Johnston Street', u'count': 51}]
```

>>>find the bank owning the most offices (office.py)

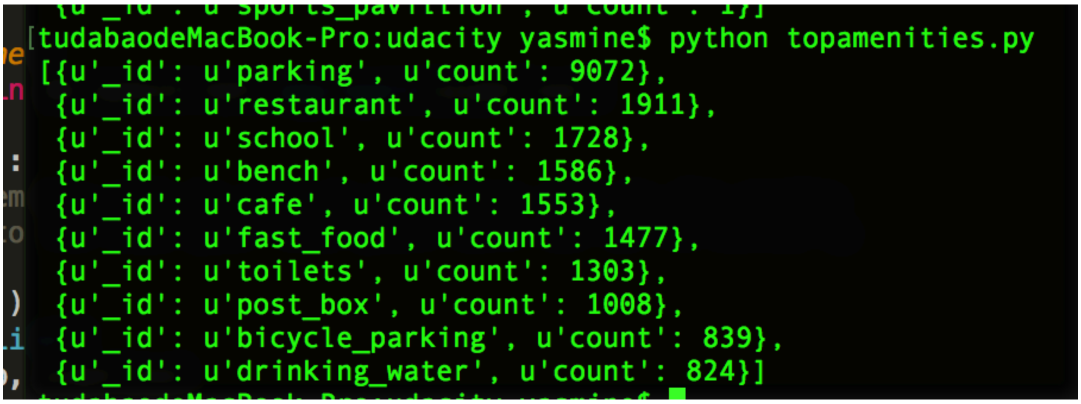
```
pipeline = [
            {"$match": {
               "amenity": "bank"}},
            {'$group': {
               "_id": "$name",
               "count": {"$sum": 1}}},
            {'$sort': {
               "count": -1}},
            {"$limit": 5}
      ]
```



```
tudabaodeMacBook-Pro:udacity yasmine$ python office.py
[{u'_id': u'Commonwealth Bank', u'count': 111},
 {u'_id': u'ANZ', u'count': 64},
 {u'_id': u'Westpac', u'count': 63},
 {u'_id': u'NAB', u'count': 57},
 {u'_id': u'Bendigo Bank', u'count': 48}]
```

>>>top 10 appearing amenities （topamenities.py)

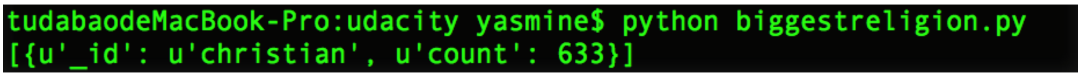
```
pipeline = [
            {"$match":{"amenity":{"$exists":1}}},
            {"$group":{"_id":"$amenity",
                     "count":{"$sum":1}}},
            {"$sort":{"count": 1}},
            {"$limit":10}
            ]
```



```
tudabaodeMacBook-Pro:udacity yasmine$ python topamenities.py
[{u'_id': u'parking', u'count': 9072},
 {u'_id': u'restaurant', u'count': 1911},
 {u'_id': u'school', u'count': 1728},
 {u'_id': u'bench', u'count': 1586},
 {u'_id': u'cafe', u'count': 1553},
 {u'_id': u'fast_food', u'count': 1477},
 {u'_id': u'toilets', u'count': 1303},
 {u'_id': u'post_box', u'count': 1008},
 {u'_id': u'bicycle_parking', u'count': 839},
 {u'_id': u'drinking_water', u'count': 824}]
```

>>>biggest religion (biggestreligion.py)

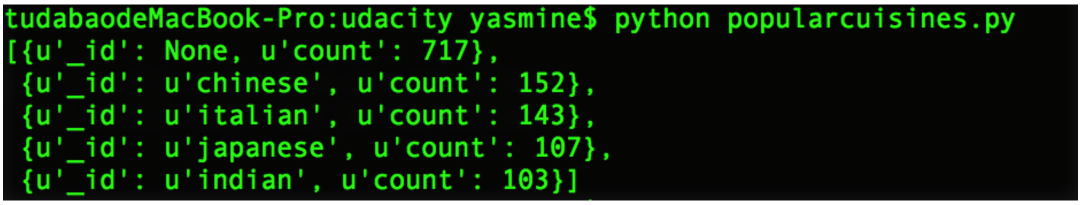
```
{"$match":{"amenity":{"$exists":1},
                  "amenity":"place_of_worship"}},
           {"$group":{"_id":"$religion",
                 "count":{"$sum":1}}},
           {"$sort":{"count": -1}},
           {"$limit":1}
```



```
tudabaodeMacBook-Pro:udacity yasmine$ python biggestreligion.py
[{u'_id': u'christian', u'count': 633}]
```

>>>most popular cuisines

```
pipeline = [
           {"$match":{"amenity":{"$exists":1},
                 "amenity":"restaurant"}},
           {"$group":{"_id":"$cuisine",
           "count":{"$sum":1}}},
           {"$sort":{"count": -1}},
           {"$limit":5}
           ]
```



```
tudabaodeMacBook-Pro:udacity yasmine$ python popularcuisines.py
[{u'_id': None, u'count': 717},
 {u'_id': u'chinese', u'count': 152},
 {u'_id': u'italian', u'count': 143},
 {u'_id': u'japanese', u'count': 107},
 {u'_id': u'indian', u'count': 103}]
```

4.conclusion

I think that the openstreet data can be joined with other kinds of data to provide more types of information, in order to allow other kinds of query.

Through the clean process, i found that the abbreviate names are not common , and the data is quite clean. Plus, if working together with a more robust data processor , it would be possible to have more cleaned data in openstreet.