OpenStreetMap Project
Data Wrangling with MongoDB

Shmuel Naaman

Map Area: Jerusalem, Israel

Problems encountered in your map

I choose to work on a section of map that includes Jerusalem the capital city of Israel. The reason for my choice is that I love, know and miss this city every day. The city was always emotional and includes so many cultures so I thought that might be an interesting challenge to go over the map database.

The first problem I encountered was that the data of the city was available only thru OpenStreetMap, so I had to download the data manually. I find out that the size of each download was limited to sections that are smaller than the size of the city. Therefore I download several sections that together include the entire city. So the next step was to write a script that merges XML files. For that I use similar script to the one we used in lesson 2 problem # 6 (processing patent), but modify it so the script merge the files instead of separate the file.

Next I check for repetitions in the file by counting repetition of the ID's field and did not find repetitions.

Last I remove all names in languages other than English or Hebrew.

```
import csv
import oss
def parse_file():
# the different files
 matching=['map_1.osm','map_2.osm','map_3.osm', 'map_4.osm',
'map_5.osm','map_6.osm',        'map_7.osm','map_8.osm']
# output file
 file_out = "{}.osm".format('jeru')
 l=0
 n = -1
# reading the different files

 for file in matching:
     with open(file,'rb') as f1:
# include or not include the first 2 lines and the last line depend on the
location of each file
# in the output file
         for line in f1:
             if '<?xml version="1.0" encoding="UTF-8"?>' in line:
                 n += 1
```

```
                if n == 0 :
                    with open(file_out,'w') as f2:
                        f2.write(line)
                else:
                    next(f1)
            else:
                if '<osm ' in line and n==0:
                    with open(file_out,'a') as f2:
                        f2.write(line)
                elif '<osm ' in line and n!=0:
                    pass
                elif '</osm>'  in line and n==(len(matching)-1):
                    with open(file_out,'a') as f2:
                        f2.write(line)
                elif '</osm>'  in line and n!=(len(matching)-1):
                    pass
                elif '<osm ' not in line and '</osm>' not in line:
                    with open(file_out,'a') as f2:
                        f2.write(line)

f2.close()
```

The data itself was different from the data we had in class, for example the address field include mostly only house number field. For example, “addres.addr:postcode” appear only 12 times.

```
print db.JERU.aggregate([     {'$group':{'_id':'$addres.addr:postcode'}},
                              {'$group':{'_id':'$count','counts':
                              {'$sum':1}}}])

{u'ok': 1.0, u'result': [{u'counts': 12, u'_id': None}]}
```

Data structure

Most of the information included in the database is descriptions of different sites, for example schools, places of worship, restaurants etc. and way of access, for example bridges, gates, barriers, etc. so I arrange the data in a different way.

```
{
  "created": {
    "changeset": "17064899",
    "user": "Son Grove",
    "version": "2",
    "uid": "963154",
    "timestamp": "2013-07-23T17:37:56Z"
  },
  "site": {
    },
  "access": {
    },
```

```
  "addres": {
    "addr:housenumber": "17",
    "addr:postcode": "93546",
    "addr:street": "Hebron Rd.",
    "addr:city": "Jerusalem",
    "addr:country": "IL"
    },
  "pos": [
    31.7526265,
    35.1899934
  ],
  "visible": "true",
  "others": {
    },
  "type": "node",
  "id": "2377341158"
}
```


“site” field include the fallowing keys

['amenity', 'brand', 'capacity', 'cuisine', 'denomination', 'designation', 'dispensing', 'fee', 'historic', 'internet_access', 'int_name', 'information', 'landuse', 'leisure', 'leader', 'military', 'material', 'man_made', 'name', 'Name', 'natural', 'operator', 'opening_hours', 'payment', 'population', 'place', 'religion', 'recycling', 'stars', 'sport', 'smoking', 'shop', 'trees', 'tourism', 'toilets', 'website']

Where 'amenity' describe the nature of the site and the other fields describes different futures include in the site.

Mangodb quarry shows the 4 top count of different 'amenity' fields in the “site” key

```
print db.JERU.aggregate([{'$group':{'_id':'$site.amenity',
                        'count':{'$sum':1}}},
                        {'$sort':{'count':-1}},
                        {'$limit':5}])

{u'ok': 1.0, u'result': [{u'count': 278999, u'_id': None},
                        {u'count': 2354, u'_id': u'school'},
                        {u'count': 2312, u'_id': u'parking'},
                        {u'count': 2044, u'_id': u'place_of_worship'},
                        {u'count': 581, u'_id': u'fuel'}]}
```

“access” field include the fallowing keys

['bus', 'bridge', 'barrier', 'bicycle', 'border_type', 'boundary', 'cycleway', 'cutting' ,'entrance', 'enforcement', 'emergency', 'electrified', 'from', 'frequency', 'foot', 'footway' ,'fence_type', 'foot', 'gauge', ,'horse', 'incline', 'junction', 'lit', 'lanes', 'motorcar', 'motor_vehicle', 'motorcycle', 'maxspeed', 'maxheight', 'network', 'public_transport', 'phone' ,'parking', 'park_ride', 'route', 'restriction', 'railway', 'surface', 'steps', 'sign', 'service', 'two_sided', 'tunnel', 'tram', 'trail_visibility', 'traffic_calming', 'tracktype', 'to', 'voltage', 'vehicle', 'width', 'wheelchair', 'waterway']

The field 'highway' was change to ‘route’ to describe different type of routes.

“access” includes fields that describe the different transportation available or ways for example bus, train, etc.

Mangodb quarry that show the 4 top counts of different ' access ' keys.

```
 print db.JERU.aggregate([     {'$group':{'_id':'$access',
                               'count':{'$sum':1}}},
                               {'$sort':{'count':-1}},
                               {'$limit':5}])

{u'ok': 1.0, u'result': [{u'count': 276349, u'_id': None},
                         {u'count': 2194, u'_id': {u'route': u'footway'}},
                         {u'count': 1819, u'_id': {u'route': u'bus_stop'}},
                         {u'count': 1487, u'_id': {u'route': u'service'}},
                         {u'count': 1316, u'_id': {u'route': u'residential'}}]}
```

Overview of the Data

Data overview

Size of the file

*OSM file : 58 MB

*JSON file : 101MB

Number of documents

print db.JERU.find().count()

288564

Number of node

print db.JERU.find({"type":"node"}).count()

260582

Number of way

print db.JERU.find({"type":"way"}).count()

27982

Number of unique users

```
print db.JERU.aggregate([      {'$group':{'_id':'$created.user'}} ,
                        {'$group':{'_id':'$count','counts':
                        {'$sum':1}}}])
{u'ok': 1.0, u'result': [{u'counts': 241, u'_id': None}]}
```

Entry count for the top 5 unique users

```
print db.JERU.aggregate([      {'$group':{'_id':'$created.user',
                        'count':{'$sum':1}}},
                        {'$sort':{'count':-1}},
                        {'$limit':5}])

{u'ok': 1.0, u'result': [      {u'count': 242032, u'_id': u'BMM994'},
                               {u'count': 10649, u'_id': u'kkl_import'},
                               {u'count': 4212, u'_id': u'eric22'},
                               {u'count': 3378, u'_id': u'ualios'},
                               {u'count': 3284, u'_id': u'Son Grove'}]}
```

We can see that the data is dominated by BMM994, where other users reach less than 5 % of his count

Problematic keys

{'lower': 52958, 'lower_colon': 30359, 'other': 2339, 'problemchars': 1}

Other ideas about the datasets

The information in this data base were reasonably organised, I did not had to modify the data very much before insert it in to mongodb. This might be due to the fact that most of the data was inserted by one user ('BMM994'). Most of the sites include in this database were educational, religious or commercial sites. Each site is accompanied with a batch of information that describes it (see the key of the "site"). The address field was only sparsely filled for example the field "addres.street" appear only 56 times. I arrange the information in a different way, I believe the way I choose to arrange the data is more suitable to represents the information in the database.