Paweł Mrozowski – Data Analyst nanodegree

pawel.mrozowski@avioaero.it

Overview of the data

The data I tried to analyze was at first my home city: Bielsko-Biala Poland. Unfortunately the data set was smaller than 50 MB so I have decided to pick another city I lived it, a lot bigger, Krakow (Cracow). Cracow has about 1 million on inhabitants and is one of the biggest cities in southern Poland

The size of the map is about 312 MB. JSON file after conversion has about 356 MB.

BASIC INFO

To know more about the map itself I have used mapparser.py. The output I have contains some statistics ;

```
{'bounds': 1, 'member': 63601, 'meta': 1, 'nd': 1616788, 'node': 1433632, 'note': 1, 'osm':
1,'relation': 2334, 'tag': 902512, 'way': 179027}
```

Problems encountered

Since the data is very clean I had problems finding what to fix, by just looking at the data.

I had to make a few queries to analyze the quality of the data.

1) Check postal codes

cleaning \postalCodes.py

OK

31- and 30 – a Krakow, some 32- are small surrounding towns

2) Check "building' property
   There may be some inconsistency since
   Some records are: 'yes', some are in fact categories like: warehouse,garages,house,office.

cleaning\ building.py

3) Check "source" property
   Although this may not be very important, this field also contains some data that are actually not useful:

   tag {'k': 'source', 'v': 'http://ump.waw.pl/ retrieved 04:46:30 05/10/10 (UMP-Krakow/src/KRAKOW_XVI_BIENCZYCE.ulice.txt)'}

   tag {'k': 'source', 'v': 'http://ump.waw.pl/ retrieved 19:07:52 04/10/10 (UMP-Krakow/src/cities-Krakow.pnt-converted.txt)'}

cleaning \ source.py

4) Check "e-mail" property
   There are probably some errors like missing letters:
   tag {'k': 'email', 'v': 'dominik.kondrat@gmai.com'}

cleaning \ emails.py

By making this kind of analysis for any other fields probably we can find other data to clean.

Queries

USER INFO

1 . listing all user id with users.py

Output: 858 users

2. query top users (quantity of records)(since we have 858 users we limited them with limit operator)

```
pipeline = [ { "$group" : {"_id" : "$created.user",
"count" : { "$sum" : 1 }}},
{ "$sort" : { "count" : -1}},
{ "$limit" : 4 }]
```

```
[{u'_id': u'W\u0142adys\u0142aw Komorek', u'count': 651536},
 {u'_id': u'ppece', u'count': 300542},
 {u'_id': u'kutomba_mfumo', u'count': 186290},
 {u'_id': u'vinci4352', u'count': 89724},
```

3. query first and last date of records

```
  pipeline = [ { "$project" : { "timestamp" : "$created.timestamp"}},
               { "$sort" : { "timestamp" : 1}},
               { "$limit" : 1 }]
```

```
 [{u'_id': ObjectId('54e6014014ff1c678c4697a9'),
  u'timestamp': u'2007-06-14T22:22:05Z'}]
```

```
  pipeline = [ { "$project" : { "timestamp" : "$created.timestamp"}},
               { "$sort" : { "timestamp" : -1}},
               { "$limit" : 1 }]
```

```
[{u'_id': ObjectId('54e6014814ff1c678c48ed01'),
  u'timestamp': u'2015-02-04T16:58:47Z'}]
```

4. query count date of records grouped by years

```
pipeline = [ { "$project" : { "datetime" : { "$substr": ["$created.timestamp", 0, 4 ]}}},
                 { "$group" : {"_id" : "$datetime",
                          "count" : { "$sum" : 1 }}},
                 { "$sort" : { "_id" : 1}}]
```

```
[{u'_id': u'2007', u'count': 17},
 {u'_id': u'2008', u'count': 14052},
 {u'_id': u'2009', u'count': 9980},
 {u'_id': u'2010', u'count': 26301},
 {u'_id': u'2011', u'count': 75019},
 {u'_id': u'2012', u'count': 320625},
 {u'_id': u'2013', u'count': 398927},
 {u'_id': u'2014', u'count': 516488},
 {u'_id': u'2015', u'count': 251250}]
```

5. query count date of records grouped by months

```
pipeline = [ { "$project" : { "datetime" : { "$substr": ["$created.timestamp", 4, 4 ]}}},
                 { "$group" : {"_id" : "$datetime",
                          "count" : { "$sum" : 1 }}},
                 { "$sort" : { "_id" : 1}}]
```

```
[{u'_id': u'-01-', u'count': 274724},
 {u'_id': u'-02-', u'count': 36560},
 {u'_id': u'-03-', u'count': 72961},
 {u'_id': u'-04-', u'count': 185368},

 {u'_id': u'-05-', u'count': 80167},

 {u'_id': u'-06-', u'count': 31488},

 {u'_id': u'-07-', u'count': 186158},

 {u'_id': u'-08-', u'count': 286663},

 {u'_id': u'-09-', u'count': 165632},

 {u'_id': u'-10-', u'count': 166707},

 {u'_id': u'-11-', u'count': 72976},

 {u'_id': u'-12-', u'count': 53255}]
```

Other ideas about the datasets

Since Kraków is very old city it is also an object for tourism. There are many places to see, and first one is Kings Palace .

To understand what are the conditions for tourism some of the queries were focused on these info:

1. How many hotels are in Cracow

```
pipeline = [ { "$project" : { "tourism" :{"$eq" :["$tourism","hotel"]} }},
             { "$group" : { "_id" : "$tourism",
                     "count" : { "$sum" : 1 }}},
             { "$sort" : { "count" : 1}}]
```

```
[{u'_id': True, u'count': 175}, {u'_id': False, u'count': 1612484}]
```

Answer is : 175

2. How many hostels are in Cracow

```
pipeline = [ { "$project" : { "tourism" :{"$eq" :["$tourism","hostel"]} }},
             { "$group" : { "_id" : "$tourism",
                     "count" : { "$sum" : 1 }}},
             { "$sort" : { "count" : 1}}]
```

```
[{u'_id': True, u'count': 83}, {u'_id': False, u'count': 1612576}]
```

Answer is : 83

3. How many information points are in Cracow
```
pipeline = [ { "$project" : { "tourism" :{"$eq" :["$tourism","information"]} }},
             { "$group" : { "_id" : "$tourism",
                     "count" : { "$sum" : 1 }}},
             { "$sort" : { "count" : 1}}]
```

```
[{u'_id': True, u'count': 92}, {u'_id': False, u'count': 1612567}]
```

Answer is : 92

4. How many restaurants there are

```
pipeline = [ {"$match" : {"amenity":"restaurant"}},

            { "$group" : { "_id" : "$amenity" ,

                    "count" : { "$sum" : 1 }}},

            { "$sort" : { "count" : -1}},

            ]
```

```
[{u'_id': u'restaurant', u'count': 377}]
```

Answer is : 377

More queries can be done on “amenity” type since *amenity* has : 104 categories

```
pipeline = [

            { "$group" : { "_id" : "$amenity" ,

                    "count" : { "$sum" : 1 }}},

            { "$sort" : { "count" : -1}},

            ]
```

Conclusion

Data aspect:

The dataset seems to be in pretty good shape, although it required to make some cleaning. In order to do this I had to investigate the db more thoroughly since this was not so obvious.

Technical aspect:

We can see that analyzing data set the size of almost 350 MB was very easy. It would not be possible or at least very difficult , with tools like Ms Excel .

From technical point of view, the process of analyze consumes small amount of CPU, but more RAM and HDD performance is needed.

Used **Sources**

http://effbot.org/zone/element-iterparse.htm

http://docs.mongodb.org/manual/reference

http://pl.python.org/kursy,jezyka.html

https://docs.python.org/2/library/xml.etree.elementtree.html