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 following queries.

Queries for basic data

Elements quantity

```
pipeline = [ { "$group" : {"_id" : "$type",
                           "count" : { "$sum" : 1 }
                           } },
             { "$sort" : { "count" : -1}}]
```

```
[{u'_id': u'node', u'count': 1433629},
 {u'_id': u'way', u'count': 178968},
 {u'_id': u'heat', u'count': 26},
 {u'_id': u'building', u'count': 22},
 {u'_id': u'public_transport+public_transport=stop_area', u'count': 6},
```

{u'_id': u'broad_leaved', u'count': 3},

{u'_id': u'parking_fee', u'count': 2},

{u'_id': u'water', u'count': 1},

{u'_id': u'indoor_range', u'count': 1},

{u'_id': u'sewage', u'count': 1}]

Users number (who created elements of a map)

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

[{u'_id': None, u'count': 837}]

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.

I have created many files to audit each data I want, since it is easier to check one by one, but actual auditing is done in one file named dataCleaningAndConversion.py.

Auditing is done in one file using case instructions. I have utilized arguments in Python sys.argv[x] and then use it as a flag parameter.

argumentToPass= sys.argv[1]

audit(element, argumentToPass):

and then we can call proper audit function like **python auditAllin1.py email**

The code is inside auditAllin1.py.

Cleaning strategy

The problems mentioned in previous point are addressed with this code:

dataCleaningAndConversion.py

procedure *cleanmyfile*

1) Check “source” property and clean non important data for 'http://ump.waw.pl/
2) Check “e-mail” property if contains @ sign and fix obviously email domains like gmai->gmail
3) Omit cities other than Cracow

file

```
for tag in element.iter("tag"):

if tag.attrib['k'] == "source":

        if 'http://ump.waw.pl/' in tag.attrib['v']:

                tag.attrib['v']='http://ump.waw.pl/'

if tag.attrib['k'] == "email": #my 2nd check, if tag's attrib is email then i check for @ sign

        if any((c in chars) for c in tag.attrib['v']):
```

```
                return node

                else:

                return None


if 'gmai' in tag.attrib['v']:

        tag.attrib['v']=tag.attrib['v'].replace("gmai", "gmail")


if tag.attrib['k'] == "addr:postcode":

        if ('31-' in tag.attrib['v']) or ('30-' in tag.attrib['v'])or ('32-' in tag.attrib['v']):

                return node

                else:

                return None
```

BEFORE CLEANING:

```
<way id="31763956" version="14" timestamp="2013-10-25T20:33:10Z" changeset="18542763"
uid="1722488" user="Mateusz Konieczny">

   <nd ref="206376392"/>

   <nd ref="595404906"/>

   <nd ref="595404908"/>

   <nd ref="2508799914"/>

   <tag k="bicycle" v="designated"/>

   <tag k="cycleway:surface" v="asphalt"/>

   <tag k="foot" v="designated"/>

   <tag k="footway:surface" v="paving_stones"/>

   <tag k="highway" v="path"/>

   <tag k="segregated" v="yes"/>

   <tag k="source" v="http://ump.waw.pl/ retrieved 11:04:55 11/17/09 (UMP-
Krakow/src/KRAKOW_XIII_PODGORZE.ulice.txt)"/>
```

<tag k="surface" v="paved"/>

</way>

AFTER CLEANING in JSON:

{"bicycle": "designated", "segregated": "yes", "created": {"uid": "1722488", "changeset": "18542763", "version": "14", "user": "Mateusz Konieczny", "timestamp": "2013-10-25T20:33:10Z"}, "surface": "paved", "source": "**http://ump.waw.pl/**", "foot": "designated", "node_refs": ["206376392", "595404906", "595404908", "2508799914"], "type": "way", "id": "31763956", "highway": "path"},

BEFORE CLEANING:

<node id="1667063823" lat="50.0260771" lon="19.9232469" version="4" timestamp="2014-06-11T19:41:48Z" changeset="22878068" uid="693154" user="Władysław Komorek">

<tag k="addr:city" v="Kraków"/>

<tag k="addr:country" v="PL"/>

<tag k="addr:housenumber" v="8a"/>

<tag k="addr:postcode" v="30-410"/>

<tag k="addr:street" v="Strąkowa"/>

<tag k="email" v="**dominik.kondrat@gmai.com"/>**

<tag k="name" v="Kwiaciarnia "Bajka""/>

<tag k="phone" v="+48 12 267-2749"/>

<tag k="shop" v="florist"/>

</node>

AFTER CLEANING in JSON:

{"shop": "florist", "name": "Kwiaciarnia \"Bajka\"", "created": {"uid": "693154", "changeset": "22878068", "version": "4", "user": "W\u0142adys\u0142aw Komorek", "timestamp": "2014-06-11T19:41:48Z"}, "pos": [50.0260771, 19.9232469], "id": "1667063823", "phone": "+48 12 267-2749", "address": {"city": "Krak\u00f3w", "street": "Str\u0105kowa", "housenumber": "8a", "postcode": "30-410", "country": "PL"}, "type": "node", "email": "**dominik.kondrat@gmail.com**"},

Other queries

1. 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},

2. 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'}]

3. 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}]

4. 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}},
            ]
```

Problems that can be resolved further: (dirty data)

-sometimes we can find comments with param “fixme”, like example below:

Question is if the data is reliable or wrong. In my opinion it depends, if creator asks for something not important or writes clearly that data is wrong, then we can delete if from analysis.

```
 <node id="2531678086" lat="49.9233194" lon="19.8064348" version="1" timestamp="2013-11-14T02:00:57Z" changeset="18886711" uid="467703" user="Zibior2">
   <tag k="addr:city" v="Radziszów"/>
   <tag k="addr:housenumber" v="92 b"/>
   <tag k="addr:postcode" v="32-052"/>
   <tag k="addr:street" v="Zawodzie"/>
   <tag k="fixme" v="popatrz na numerek (trzeba usunąć spację?)"/>
   <tag k="source:addr" v="EMUiA (emuia.geoportal.gov.pl)"/>
 </node>
```

```
<tag k="fixme" v="verify onewayness (it is passable south to north, used to be one-way with passage impossible in this direction)"/>
```

-“source” attribute sometimes contains non relevant data, like

<tag k="source" v="http://ump.waw.pl/ retrieved 04:46:30 05/10/10 (UMP-Krakow/src/KRAKOW_XV_MISTRZEJOWICE.ulice.txt)"/>

If we want to group data to understand who is the biggest contributor in terms of data source we can delete non important timestamp and leave only www address.

Information that we can extract, that can be used to plan the visit:

-where potential tourist can stay :

hotels, guest-house, hostel, camp_site

-what can be interesting and gather some statistics grouped by postal code(sometimes surrounding small cities can be more interesting)

viewpoint, museum (a lot),artwork,gallery (a lot)

-what are the most interesting streets (like the one close to central square) and what kind of attractions there are (amenities like pubs, restaurants), so using data grouping (can be used to plan visit)

Szewska Street, Karmelicka Street

Trying to find proper standard we may ask if the pace we plan to stay offers internet_access or no.

-sometimes there are certain historical places to visit, not related to leisure activity only ,in our case we have a few described as

<tag k="historic" v=" "/>

The output is “castle” and “cemetery”.

-visitors can also benefit from culture places like libraries,

Can be found be searching amenities types like ‘biblioteka’ and since Krakow is very old city some of them are actually historical places, rather than modern one, so can be also good places for tourist.

If the data set is bigger and contains more details, users would be able to ask questions about:

Proper standards about hotels (prices, wifi, hotel types, safe yes/no, parking place secured yes/no, additional leisure activity like swimming pool, gym)

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