Wrangle_Openstreetmap_Data

March 7, 2016

1 Wrangling Openstreetmap Data

1.1 1. Introduction

Choose any area of the world from https://www.openstreetmap.org , and download a XML OSM dataset. The dataset should be at least 50MB in size (uncompressed).

Use the Overpass API to download a custom square area. Explanation of the syntax can found in the wiki . In general you will want to use the following query:

```
(node(minimum_latitude, minimum_longitude, maximum_latitude, maximum_longitude);<;);out meta;
```

e.g.

```
(node(51.249,7.148,51.251,7.152);<;);out meta;
```

the meta option is included so the elements contain timestamp and user information. You can use the Open Street Map Export Tool to find the coordinates of your bounding box. Note: You will not be able to use the Export Tool to actually download the data, the area required for this project is too large.

1.1.1 1.1 Selected map area

The selected area to work with is all the area near **Santander - Spain**, given by this bounding box query:

```
<osm-script>
  <union into="_">
    <bbox-query e="-3.7086" into="_" n="43.4988" s="43.3496" w="-4.0574"/>
    <recurse from="_" into="_" type="up"/>
  </union>
  <print e="" from="_" geometry="skeleton" limit="" mode="meta" n="" order="id" s="" w=""/>
</osm-script>
```

1.2 2. Problems encountered in the map

Before parsing the whole dataset and using the utility provided in the assignment and uploaded in the code repository https://github.com/jmonterrubio/Wrangle_Openstreetmap_Data/blob/master/map_parser/sampler.py, we create a sample dataset to audit it.

A first look of the dataset reveals that the data is very consistent. Tipical problems like postal code, street address or email doesn't seems to have problems.

The main problems found while processing the data are:

- Undefined keys
- Inconsistent is_in field
- Inconsistent phone numbers (+34 647 724613, +34 630164635)
- Bad formatted websites (http:\\www.cantabria.es , www.Rolisas.es, ww.cibermacu.com)

1.2.1 2.1 Fixing problems

2.1.1 Undefined keys The way we fix this is ommit them in the map parse process.

https://github.com/jmonterrubio/Wrangle_Openstreetmap_Data/blob/master/map_parser/map/tags.py#L15

2.1.2 is_in field hasn't a consistent format We can see that 'is_in' values have different formats. A list of things that may be fixed are: * Use semi-colons to separate entities intead of commas * Use same language (in this case for consistency, everything in spanish) * Add well known entities (España, Europa) if missed. * Remove duplicated entities * Order the entities from smaller to bigger

Before trying to fix them, lets look for more information about the key: http://wiki.openstreetmap.org/wiki/Key:is_in. We can see there isn't a mandatory way to fill the field, so lets implement the field fix.

In this case, the code implementing this fixes could be found at https://github.com/jmonterrubio/Wrangle_Openstreetmap_Data/blob/master/map_parser/map/region.py

1.2.2 2.2 Importing data

Once the data is audited and cleaned we can parse it. The idea is to transform the **XML** format of the input file to a **JSON** format, interpretable and ready to import into MongoDB.

This code could be find in Wrangle_Openstreetmap_Data repository from my github account.

Running the code selecting an **OSM** file as the input will result in a **JSON** file as output. Importing a JSON file into mongoDB is as easy as executing the command (note that you must have mongodb installed):

```
mongoimport --db udacityP3 --collection near_santander --file near_santander.osm.json
```

1.3 3. Overview of the Data

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

- File sizes

```
near_santander.osm ......... 76 MB
near_santander.osm.json .... 82 MB
```

- Number of documents

```
> db.near_santander.count()
354970
```

- Number of nodes

```
> db.near_santander.count({"type":"node"})
319053
```

- Number of ways

```
> db.near_santander.count({"type":"way"})
35874
```

- Number of unique users

```
> db.near_santander.distinct("created.user").length
251
```

- Top 1 contributing user

```
> db.near_santander.aggregate([
    {"$group":
        {"_id":"$created.user", "count":{"$sum":1}}
    },
    {"$sort":
        {"count":-1}
    },
    {"$limit":1}
])
{"_id" : "Emilio Gomez", "count" : 164316}
```

- Number of users appearing only once (having 1 post)

```
> db.near_santander.aggregate([
    {"$group":
        {"_id":"$created.user", "count":{"$sum":1}}
    },
    {"$group":
        {"_id":"$count", "num_users":{"$sum":1}}
    },
    {"$match":
        {"_id":1}
    }
])
{"_id" : 1, "num_users" : 62}
```

- Number of different keys

```
> db.near_santander.mapReduce(
  function() {
    for (var key in this) { emit(key, 1); }
  },
  function(key, n) {
      return Array.sum(n);
  },
  {out: "near_santander_keys"}
)
> db.near_santander_keys.count()
479
```

- Top 10 used key

```
> db.near_santander_keys.find(
    {"_id":
        {$nin:["_id", "id", "created", "type"]}
    }).
    sort({"value":-1})
{"_id" : "pos", "value" : 319056}
{"_id" : "node_refs", "value" : 35914}
{"_id" : "source", "value" : 30892}
{"_id" : "highway", "value" : 22134}
{"_id" : "name", "value" : 14008}
{"_id" : "building", "value" : 13552}
{"_id" : "created_by", "value" : 10932}
```

```
{"_id" : "source:date", "value" : 10891}
{"_id" : "building:levels", "value" : 8689}
{"_id" : "address", "value" : 8491}
```

- Number of keys used only once

```
> db.near_santander_key.count({value:1})
111
```

- Top 10 amenities

```
> db.near_santander.aggregate([
    {"$match":
        {"amenity":{"$exists":1}}
    },
    {"$group":
        {"_id":"$amenity",
         "count":{"$sum":1}}
    },
    {"$sort":
        {"count":-1}
    },
    {"$limit":10}
])
{"_id" : "recycling", "count" : 322}
{"_id" : "parking", "count" : 277}
{"_id" : "restaurant", "count" : 181}
{"_id" : "place_of_worship", "count" : 168}
{"_id" : "drinking_water", "count" : 136}
{"_id" : "bench", "count" : 132}
{"_id" : "school", "count" : 110}
{"_id" : "parking_space", "count" : 106}
{"_id" : "bar", "count" : 94}
{"_id" : "waste_disposal", "count" : 91}
```

1.4 4. Other ideas about the datasets

Some ideas come to my mind when i see this dataset. Firt of all is due to the lack of information and few maintenance something must be done. For example this population was too many years ago:

```
{
    "_id" : ObjectId("56d1798800a9a619ba0e29ef"),
    "source:name" : "Nomenclátor Geográfico de Municipios y Entidades de Población",
    "source:file" : "http://centrodedescargas.cnig.es/CentroDescargas/equipamiento/BD_Municipios-Entida
    "source:ele" : "MDT5",
    "name" : "Pámanes",
    "created" : {
        "changeset" : "9515624",
        "user" : "egrn",
        "version" : "2",
        "uid" : "399394",
        "timestamp" : "2011-10-09T18:25:42Z"
    },
    "ref:ine" : "39037000200",
    "pos" : [
        43.3566433,
        -3.773787
    ],
    "ele" : "84",
    "source:date" : "2011-06",
    "source" : "Instituto Geográfico Nacional",
    "place" : "suburb",
    "population:date" : "2009",
    "is_in:province_code" : "39",
    "type" : "node",
    "id" : "259168943",
    "population" : "749"
}
```

This kind of things made the data less powerful. So why not rewarding people that set and update the data. Private amenities like restaurants or even the city halls could give some money to mantain it.

The benefit of this improvement is a better dataset, with more and most up-to-date info. It's not easy to define a good way set a budget. For example, distributing a final budget between all the contributors. Or an small amount for each update.

A problem here is about bad data, and how can you deal with users that set wrong data just for earning some money.

Another thing that could be done with the dataset is trying to know some things about the cities. For example:

- Is a poor or a rich city?
- Age average
- Tourism level

All this questions maybe could be answered using machine learning techniques that i expect to learn during this udacity nanodegree program.

The benefit is very clear, classify cities using data could help search engines of travel or real estate agencies to know more about a destiny.

The main problem maybe is the lack of information in some cases that could create some inaccuracy in the classification or clustering problems.

1.5 5. Appendix

All the code used to process the openstreetmap data is stored in my github account https://github.com/jmonterrubio/Wrangle_Openstreetmap_Data.

MongoDB version used to explore the data is v3.2.1 used with a management tool called Robomongo.