Project 3 - Wrangle OpenStreetMap Data¶

Michail Kovanis

Introduction

OpenStreetMap.org is an open source online map service. It uses tags to represent physical features on the ground. Its structure consists of nodes, ways and relations.

The nodes represent single points in the map and are defined by their latitude, longitude and id. The ways are ordered lists of nodes (2 to 2,000). The relations define logical or geographical relationships between other elements. They are ordered lists of one or more nodes, ways and/ or relations.

The nodes, ways and relations usually have at least one tag to define their purpose.

Example of a node:

```
< node id="25496583" lat="51.5173639" lon="-0.140043" version="1"
  changeset="203496" user="80n" uid="1238" visible="true"
  timestamp="2007-01-28T11:40:26Z" >
        < tag k="highway" v="traffic_signals"/ >
< /node >
```

Example of a way:

```
< way id="5090250" visible="true" timestamp="2009-01-19T19:07:25Z"
  version="8"      changeset="816806" user="Blumpsy" uid="64226" >
        < nd ref="822403"/>
        < nd ref="21533912"/>
        < nd ref="821601"/>
        < nd ref="21533910"/>
        < nd ref="135791608"/>
        < nd ref="333725784"/>
        < nd ref="333725781"/>
        < nd ref="333725774"/>
        < nd ref="333725776"/>
        < nd ref="823771"/>
        < tag k="highway" v="residential"/>
        < tag k="name" v="Clipstone Street"/>
        < tag k="oneway" v="yes"/>
< /way >
```

Example of a relation:

```
< relation id="1" >
        < tag k="type" v="boundary" />
```

```
        < tag k="boundary" v="administrative" />
        < tag k="land_area" v="administrative" />
        < tag k="admin_level" v="2" />
        < tag k="name" v="light green country" />
        < member type="way" id="AB" role="outer" />
        < member type="way" id="AC" role="inner" />

< /relation >

< relation id="2" >
        < tag k="type" v="boundary" />
        < tag k="boundary" v="administrative" />
        < tag k="land_area" v="administrative" />
        < tag k="admin_level" v="2" />
        < tag k="name" v="dark green country" />
        < member type="way" id="AB" role="outer" />
        < member type="way" id="AC" role="outer" />
< /relation>
```

Methods

In this project, I explored the OpenStreetMap data for the city of Athens, Greece (downloaded from https://mapzen.com/data/metro-extracts/athens_greece/). The uncompressed size of the OSM file was 468.6 MB. I followed the methodology and code as presented in the respective case study for the MongoDB of the "Wrangle OpenStreetMap Data" course.

I used the ElementTree library of python to parse the xml file which contained the raw data. Since the data sometimes were presented in Greek characters, I converted them to latin characters when necessary.

Finally, I passed the data to a MongoDB database, to which I performed certain queries to uncover unique contributors, cities amenities etc. included in the data.

Results

Here I present in a descriptive way the results I obtained through wrangling and analyzing the dataset. All code and analytical outputs are in the respective Jupyter notebook file (and in its .html version).

In this dataset, there are 2,198,514 nodes, 266,412 ways and 7,522 relations tags. Auditing the ‘k’ values of the tags I found that most of them were in an expected format (either lowercase or lowercase with a colon inside), only 7 of them had some sort of special characters and 3,382 were in an "other" format (probably containing Greek characters). Moreover, I identified that there were 1,362 unique contributors to this dataset.

Improving street names

The names of Greek streets very often start with the street type. Thus, I split the code to check for street type also in the beginning of the street name. Furthermore, I added some common Greek street names as "Odos" and "Leoforos" (which translate to "Street" and "Avenue", respectively). When encountering these terms, I also put in parenthesis their translation. Also, I converted all street names which were in Greek characters to their equivalent latin characters.

Some of the street names are given both in Greek and English characters, for example Τζων Κένεντυ (John Kennedy). These cases were treated as all the rest (modified to "Tzon Kenentu (John Kennedy)"). Note that the Greek character "υ" when alone is pronounced as the letter "e", but when after 'o' and 'α' can be pronounced in different ways ("ou", "av" and "af" are some of them).

Certain Greek letters are equivalent to double characters in English, such as 'θ' which is pronounced as 'th'. However, in typical Greek to "Greeklish" converters it is represented as an "8". This is understandable to Greeks, but probably not to non-Greeks. I modified a bit the code that I used to convert Greek to latin characters in order to make it easier to pronounce from non-Greeks.

Some street names are obviously street numbers. These were marked as "Invalid Street Name" when encountered.

A few of the street names also have their first character as lowercase. I converted all first characters to capitals for nicer presentation.

The other fields in which I identified Greek characters are:

1. 'name':'el'
2. 'wikipedia':'el'
3. 'created':'user'
4. 'alt_name':
5. 'name':
6. 'addr':'city'

Moreover, some of the street names are duplicates. For example, "Leoforos Pasidonos Avenue" is the same as "Poseidonos Avenue" and "Liosion (Liosion)" is the same as "Liossion". I chose not to account for duplicates in this project.

Checking for correct postal codes

Greek postal codes are in a 5-digit format, for example 13050. Here I made sure that all postal codes are in this format. If there was a space in the postal code, then I removed it and if there were more or less than 5 digits I marked it as “Invalid Postal Code”. I marked a postal code as Invalid if there was also a letter in it.

Importing to the database

I converted the data from xml to a python dictionary to pass it to the MongoBD. Moreover, after looking multiple times at the dictionary file produced from the "Preparing for database" section I found other fields which have unicode characters.

In some of the fields there are Russian or French characters. For this project, I chose to change only the Greek characters to latin and to not account for the rest. The methods 'convert()' and 'get_conversion_pool()' that are defined in this section were used to make the conversion. An example of the resulting dictionary can be seen below.


```
{'created': {'changeset': '11380916',
   'timestamp': '2012-04-22T07:49:12Z',
   'uid': '653',
   'user': 'aitolos',
   'version': '8'},
  'id': '242137',
  'pos': [37.9748947, 23.7264948],
  'type': 'node'}

 {'admin_level': '2',
  'alt_name': {'vi': u'A-Ten;Nh\xe3 \u0110i\u1ec3n'},
  'capital': 'Yes',
  'capital_ISO3166-1': 'Yes',
  'created': {'changeset': '41877248',
   'timestamp': '2016-09-02T20:29:32Z',
   'uid': '52087',
   'user': 'kocio',
   'version': '76'},
  'ele': '70',
  'id': '441183',
  'int_name': 'Athens',
  'is_in': {'country': 'Greece'},
  'name': {'zu': 'I-Athene'},
  'name:bat-smg': u'Atien\u0101',
  'name:be-tarask': u'\u0410\u0442\u044d\u043d\u044b',
  'name:fiu-vro': 'Ateena',
  'name:roa-rup': 'Athina',
  'name:roa-tara': 'Atene',
  'name:sr-Latn': 'Atina',
  'name:zh-min-nan': u'Ath\xedna',
  'name:zh-yue': u'\u96c5\u5178',
  'old_name': {'ca': 'Cetines'},
  'place': 'City',
  'population': '3090508',
  'pos': [37.9841493, 23.7279843],
  'type': 'node',
  'wikipedia': u'El:Athina'}

 {'created': {'changeset': '30148875',
   'timestamp': '2015-04-11T18:32:29Z',
   'uid': '2365615',
   'user': 'asyr',
   'version': '3'},
  'id': '1566291',
```

```
'pos': [37.5140915, 23.4425079],
'type': 'node'}
```

The database contains 2,464,926 data points, which is the sum of ‘nodes’ and ‘ways’. Querying the database for an example data point we obtain:

Query:

```
db.athens_map.find_one()
```

Result:

```
{u'_id': ObjectId('583ee56a8a99fb16002d0bef'),
 u'created': {u'changeset': u'10662021',
  u'timestamp': u'2012-02-12T11:51:59Z',
  u'uid': u'497621',
  u'user': u'NikosSkouteris',
  u'version': u'5'},
 u'id': u'78695',
 u'pos': [37.5964742, 23.0709818],
 u'type': u'node'}
```

Unique contributors

we saw that the total number of unique contributors is 1,362. What is very interesting is that the top contributing user has contributed more than the 10 next top contributors. With a simple look at the results we see that there is a strong imbalance in the contribution effort. Below we can see the top 11 contributors.

Query:

```
user = db.athens_map.aggregate([{'$group': {"_id": '$created.user',
                                            "count": {"$sum": 1}}},
                                {'$sort': {"count": -1}},
                                {"$limit":50}])
```

Result:

```
1161365  => makmar
372124  => greecemapper
134356  => mtraveller
111571  => Amaroussi
77301  => NikosSkouteris
63682  => Chris Makridis
61411  => aitolos
54137  => Kanenas
30202  => athinaios
29896  => AiNikolas
21382 => GeorgeKM
```

The ‘city’ field

I grouped all occurrences of a same city field that are found inside the data and sorted it in descending order.

The first issue I encountered is that there are 2,449,144 data points without the city tag. This is more than 99% of the data points and shows that a lot of work is necessary for the data concerning the area of Athens in the openstreetmap.org.

The rest of the results might be a bit strange for those not familiar with the city. Athens and Pireus are the two major cities in the area and the rest are suburbs and municipalities which are usually referred as parts of the closest major city. Also, some of the results are parts of the greater are called 'Attiki' or 'Attica', but it is ok to consider most of them as suburbs of Athens.

The results for 'Athina', 'Athens', 'Athen' and 'Athina (Athens)' refer to Athens. 'Athina - Alimos' refers to the area of Alimos in Athens. Almost all the rest of the results are quite normal in their majority, apart from 'Thiva' and 'Marmari', which are not at all in the area of Athens, but are inside the dataset probably due to a mistake.

Very notable is that the suburb of 'Acharnes' (3rd place) appears more than 'Peiraias' (4th place). It is an odd result because the latter is bigger and more populated than the former. The most reasonable explanation is that some of the contributors might be from 'Acharnes' and thus they worked more in documenting that area.

Here I present the first top 10 cities (by number of appearances). The rest can be found in the notebook file (or its accompanying html).

Query:

```
city = db.athens_map.aggregate([{'$group': {"_id": '$address.city',
                                            "count": {"$sum": 1}}},
                                {'$sort': {"count": -1}}])
```

Result:

2449144 => None
9873 => Athina
1783 => Acharnes
1628 => Peiraias
556 => Kallithea
235 => Arguroupoli
134 => Marousi
93 => Moschato
83 => Galatsi
72 => Nea Ionia

Counting the top amenities

In this part I counted all the unique occurrences of different types of amenities. From the results, one can note that cafeterias are very frequent. This is actually true for Athens and it shouldn't be a surprise.

A very surprising result is the number of places of worship. In the area of Attica only greek-orthodox, and very few other Christian, churches exist. There are no mosques. A search on greek websites shows that in the area of Attica exist 960 greek-orthodox churches. The data suggests that $1,016$ places of worship exist. This means that actually this result is very accurate in the OpenStreetMap for Athens. Some of the other results to me seem to be a bit under-documented, especially those for 'Atm', 'Bar', 'Fast_Food' and 'School'. Interestingly archaeological spaces are missing from the top 30 and this also doesn't seem normal. Here I present the first top 10 amenities (by number of appearances). The rest can be found in the notebook file (or its accompanying html).

Query:

```
amenity = db.athens_map.aggregate([ {"$match":{"amenity":{"$exists":1}}},
                                    {'$group': {"_id": '$amenity',
                                                "count": {"$sum": 1}}},
                                    {'$sort': {"count": -1}},
                                    {"$limit":30}])
```

Result:

1650 => Bench
1296 => Cafe
1266 => Parking
1137 => Restaurant
1016 => Place_Of_Worship
986 => School
771 => Pharmacy
725 => Fuel
673 => Fast_Food
506 => Bank

Discussion

In this project, I analyzed the dataset of OpenStreetMap for the area of Athens. I parsed the xml file, I cleaned it to an extent, formatted it into a python dictionary and passed into a MongoDB.

Even though I converted all the Greek characters to latin, this is not enough. There are data points that include Russian, French and other characters which also need conversion to latin.

From the queries, I performed in the database seems that a lot of additional work needs to be done in documenting the "city" field in the data points, especially to reduce double counting of the same areas. For that purpose, there should be a decision whether areas should be indexed with only Greek characters (e.g. "Αθήνα"), both Greek and English characters (e.g. "Αθήνα (Athina)"), both Greek and English names (e.g. "Αθήνα (Athens)") or only in English (e.g. "Athens"). Also, the various areas and suburbs can be either presented with only their name (e.g. "Kifisia") or

both with their name and the name of the bigger city or area closer to them (e.g. "Kifisia (Athens)").

In my opinion the best way to represent such data is in the following format:

Κηφησιά - Αθήνα (Kifisia - Athens)

There should be a document with sample name-types for all the required fields, so that all contributors can follow and thus making the dataset cleaner. For instance, a model representation for an amenity could be:

<tag k="addr:city" v=" Κηφησιά - Αθήνα (Kifisia - Athens)"/>

<tag k="addr:housenumber" v="10"/> *(lowercase letters only at the end)*

<tag k="addr:postcode" v="13050"/> *(5 numbers without spaces)*

<tag k="addr:street" v="Οδός Κηφησίας - Odos (Street) Kifisias"/>

<tag k="amenity" v="Εστιατόριο Restaurant"/> *(both in Greek and English)*

<tag k="cuisine" v="Ελληνική - Greek "/> *(both in Greek and English)*

<tag k="name" v="Πάργα - Parga"/> *(both in Greek and Latin characters)*

<tag k="outdoor_seating" v="Ναι - Yes"/> *(yes or no both in Greek and English)*

<tag k="phone" v="2105736123"/> *(10 numbers without spaces)*

<tag k="smoking" v=" Ναι - Yes "/> *(yes or no both in Greek and English)*

<tag k="takeaway" v=" Ναι - Yes "/> *(yes or no both in Greek and English)*

Finally, there is a strong imbalance in the contributions by the users. This means that very few people are responsible for most of the work in the dataset. In my opinion this is a major reason for having many incomplete data points and an over-representation of some areas, such as 'Acharnes'.

References

OpenStreetMap information: https://www.openstreetmap.org/

OpenStreetMap wiki: https://wiki.openstreetmap.org/wiki/Main_Page

get_element() method of the xml parser: http://stackoverflow.com/questions/3095434/inserting-newlines-in-xml-file-generated-via-xml-etree-elementtree-in-python

Python documentation: https://www.python.org/doc/

Pymongo documentation: https://api.mongodb.com/python/current/

Greek to latin converter: https://www.g-loaded.eu/2006/12/18/pygr2gl-Greek-to-Greeklish-converter/

Count digits in string: https://stackoverflow.com/questions/24878174/how-to-count-digits-letters-spaces-for-a-string-in-python

Churches in Attica: http://parganews.com/%CF%80%CF%8C%CF%83%CE%B5%CF%82-%CE%B5%CE%BA%CE%BA%CE%BB%CE%B7%CF%83%CE%AF%CE%B5%CF%82-%CF%85%CF%80%CE%AC%CF%81%CF%87%CE%BF%CF%85%CE%BD-%CF%83%CF%84%CE%B7%CE%BD-%CF%85%CF%80%CE%AD%CF%81%CE%BF%CF%87/