OpenStreetMap Data Case Study

Map Area

Bucharest, Romania

```
https://www.openstreetmap.org/way/22090768#map=12/44.4389/26.0960
https://mapzen.com/data/metro-extracts/metro/bucharest_romania/
```

For this project, I am going to audit and clean the map exported from OpenStreetMap for Bucharest, Romania. I am curious about the problems and inconsistencies I'm able to discover in this data. Then, I'm going to run few SQL queries on the cleaned data, one of my topic of interest being related to the mobile shops of the main telecom companies present on the Romanian market: Vodafone, Orange and Telekom.

Problems Encountered in the Map

1. The street names are abbreviated inconsistently
2. Some of the street numbers contain the street name, others contains irrelevant information and no number (I coded these with "No_number").
3. The mobile shops names has the partner's name associated. I cleaned the name t show only the name of the mobile operator.

Cleaning of Street Names

There are street names with their types abbreviated. In Romania, the street type appears at the beginning of the street name. I want to have a consistancy in the street type, so for example "sos" was relabeled to "Soseaua", "Str" to "Strada", etc.

In [ ]:

```
### Mapping correct street type
mapping_street = {"sos": "Șoseaua",
            "Spaiul.": "Splaiul",
            "Str.":"Strada",
            "Piata":u"Piața",
            "Dr.":"Drumul",
            "Drum":"Drumul",
            "drumul":"Drumul",
            "Bulevbardul":"Bulevardul",
            "B-dul":"Bulevardul",
            "strada":"Strada",
            "Soseaua":u"Șoseaua",
```

```
            "Spaiul":"Splaiul",
            "strada":"Strada",
            "Sf.":"Sfantul",
            "pictor":"Pictor",
            "Campia":u"Câmpia"
           }
#Update the street names
def update_street(name, mapping_street):
    m = street_type_re.search(name)
    better_name = name
    ##### condition: if the street name does have a last word
    if m:
        # check if the street type is a key in your mapping dictionary:
        if m.group() in mapping_street.keys():
            better_street_type = mapping_street[m.group()]
            better_name = street_type_re.sub(better_street_type, name)
        return better_name
```

Cleaning the shops names

To find the mobile shops I have to check two tags: one contains the label "mobile_phone" which flag it as a mobile shop, the other returns the name of the shop. This is how it looks like:

In [ ]:

```
<tag k="name" v="Germanos Băneasa"/>
<tag k="shop" v="mobile_phone"/>
```

Some tag's values contains not only the mobile operator: Vodafone, Telekom or Orange, but also the name of their partners like: Internity, Telekom's partner or Arsis, which is Vodafone's partner. Also, some Telekom shops are labeled with the name Cosmote. Since 2014, Cosmote was rebranded as Telekom.

In [ ]:

```
### Mapping correct shops names
mapping_shop = {"Arsis Vodafone": "Vodafone",
            "Cosmote / Romtelecom": "Telekom",
            "Interniti, Cosmote": "Telekom",
            "Cosmote": "Telekom",
            "Interniti, Cosmote ": "Telekom",
            "Germanos":"Telekom",
            u"Germanos Băneasa":"Telekom",
            "Germanos Promenada":"Telekom",
            "Ilex Vodafone":"Vodafone",
            "Orange Shop":"Orange",
            "Orange Store":"Orange",
            "Orange Titan":"Orange",
            "Telekom Titan":"Telekom",
            "Vodafone"+' "ILEX"':"Vodafone",
            "Vodafone Arsis":"Vodafone",
            "Vodafone Titan":"Vodafone",
            "Vodafone Vasile Milea":"Vodafone"
           }
#Update the shops names
def update_mobile_shop(name, mapping_shop):
    for key,value in mapping_shop.items():
```

```
        if name==key:
            name=value
        else:
            continue
    return name
```

Cleaning the streets numbers

In Bucharest, the street numbers should be a number or a number followed by a letter or a range of letters. After checking the streets numbers, I noticed that some of them contain wrong information, like: -the street name:"Soseaua Virtutii". I re-labeled it to "No_number" -the street name and the street name: "B-dul. Iuliu Maniu, nr.79, Sector 6". I extracted only the number. -the number of the block of flats: "Bl. 43", "Bloc F5". I re-labeled it to "No_number" Also, I remove whitespaces and convert the street number to uppercases

In [ ]:

```
### Mapping correct streets numbers
mapping_housenumber = { "B-dul. Iuliu Maniu, nr.79, Sector 6": "79",
            "185Miniprix": "185",
            "Bl. 43": "No_number",
            "Bl. 50": "No_number",
            "Bl. 37": "No_number",
            "Bl. 38": "No_number",
            "106R.  Can Pack": "106R",
            "Bl. PM91":"No_number",
            "Sos. Pipera-Tunari, nr.1C, Sector 2":"1C",
            "Brasov":"No_number",
            "F.N.":"No_number",
            "37/I":"37I",
            "31B+C":"31B,C",
            "Bl. 5":"No_number",
            "Bl. 4":"No_number",
            "Bl. 7":"No_number",
            "Bl. 1":"No_number",
            "Bl. 3":"No_number",
            "Bl. M15":"No_number",
            "37/J":"37J",
            "486Petrom":"486",
            "4, Ap.2, Sector 5":"4",
            "bl. 1":"No_number",
            "bl. 2":"No_number",
            "Bl. P2":"No_number",
            "1A villa 40":"No_number",
            "Bloc F5":"No_number",
            "Bloc C5":"No_number",
            "Soseaua Virtutii":"No_number",
            "Nr. 232":"232",
            "Bl. 12":"No_number",
            "nr.130":"130",
            "1A (fost 14)":"1A",
            "Cămin 4":"No_number",
            "86     Bl 86A":"86",
            "nr. 12":"12",
            "km9+100":"No_number",
            "Scara C":"No_number",
            "Bl 303":"No_number",
            "a1 Et4 Sc a":"A1"
```

```
            "Pădurea Băneasa":"No_number",
            "5 ,bl.C 72":"5",
            "Bl. C3":"No_number",
            "Bl. C2":"No_number",
            "Bl. C1":"No_number",
            "fn":"No_number",
            "BL.P 20":"No_number",
            "Bl.1":"No_number",
            "Bl. A5":"No_number"
    }

remove_whitespaces = re.compile(r'\s+')           #regular expression to
remove whitespaces

def update_number(number, mapping_housenumber):
    number=number.encode('utf-8')
    for key,value in mapping_housenumber.items(): #update housenumbers with
values from mapping
        if number==key:
            number=value
    number=re.sub(remove_whitespaces, '',number)  #remove whitespaces and c
onvert the housenumber to uppercases
    number=str.upper(number)
    return  number
```

Data Overview and Additional Ideas

This section contains basic statistics about the dataset, the SQL queries used to gather them, and some additional ideas about the data in context.

Number of nodes

In [ ]:

```
QUERY = "select COUNT(*) from nodes "
cur.execute(QUERY)
result = cur.fetchone()
print "Number of nodes:", result[0]
```

Number of nodes: 169402

Number of ways

In [ ]:

```
QUERY = "select COUNT(*) from ways "
cur.execute(QUERY)
result = cur.fetchone()
print "Number of ways:", result[0]
```

Number of ways: 133243

Number of unique users

Number of unique users

In [ ]:

```
QUERY = "select count(distinct uid) from (select uid from nodes union all s
elect uid from ways)"
cur.execute(QUERY)
result = cur.fetchone()
print "Number of unique users:", result[0]
```

In [ ]:

```
Number of unique users: 1291
```

Top 3 contributing users

In [ ]:

```
QUERY = "SELECT user, count(*) as USERS_COUNT \
         FROM (select user, uid from nodes UNION ALL select user, uid FROM
ways) \
         GROUP BY user \
         ORDER BY USERS_COUNT DESC \
         LIMIT 3"
cur.execute(QUERY)
result = cur.fetchall()
print "----top 3  contributing users----"
c=1
for i in result:
    print c,"User name: %s, Posts counts: %s" % (i[0], i[1])
    c+=1
```

In [ ]:

```
----top 3  contributing users----
1 User name: razor74, Posts counts: 101131
2 User name: Stereo_repair, Posts counts: 60206
3 User name: Strainubot, Posts counts: 40727
```

Number of users contributing once

In [ ]:

```
QUERY = "SELECT count(*) FROM \
         (SELECT user, count(*) as USERS_COUNT \
         FROM (select user, uid from nodes UNION ALL select user, uid FROM
ways) \
         GROUP BY user \
         HAVING USERS_COUNT=1)"
cur.execute(QUERY)
result = cur.fetchone()
print "----Users contributing once----"
print "Number of users contributing once:", result[0]
```

In [ ]:

```
----Users contributing once----
Number of users contributing once: 521
```

Total number of contributions

In [ ]:

```
QUERY = "SELECT count(*) \
         FROM (select user, uid from nodes UNION ALL select user, uid FROM
ways) \
         GROUP BY user"

cur.execute(QUERY)
result = cur.fetchall()
```

In [ ]:

```
#create a liste from the tuple result and sum its elements to find the
number of total contributions
lista=[]
for i in range(len(result)):
    lista.append(result[i][0])

contributions=0
for i in lista:
    contributions+=i

print "----Total  numbers of contributions----"
print contributions
```

In [ ]:

```
----Total  numbers of contributions----
302645
```

Number of mobile shops

In [ ]:

```
QUERY = "SELECT value, count(*)\
         FROM nodes_tags WHERE  value in ('Telekom','Orange','Vodafone') \
         GROUP BY value"
cur.execute(QUERY)
result = cur.fetchall()
print "----Mobile shops----"
for i in result:
    print "Shop name: %s, counts: %s" % (i[0], i[1])
```

In [ ]:

```
----Mobile shops----
Shop name: Orange, counts: 18
Shop name: Telekom, counts: 19
Shop name: Vodafone, counts: 21
```

Operator with most shops

In [ ]:

```
QUERY = "SELECT value, max(No_shops) FROM \
          (SELECT value, count(*) as No_shops\
           FROM nodes_tags WHERE value IN ('Telekom','Orange','Vodafone') \
           GROUP BY value)"
cur.execute(QUERY)
result = cur.fetchall()
print "----Mobile operator with most shops----"
for i in result:
    print "Shop name: %s, max counts: %s" % (i[0], i[1])
```

In [ ]:

```
----Mobile operator with most shops----
Shop name: Vodafone, max counts: 21
```

Shops position on map

In [ ]:

```
QUERY = "SELECT value, lon, lat FROM (\
        (SELECT id, value  \
            FROM nodes_tags WHERE value IN ('Telekom','Orange','Vodafone'))
t_shops \
          LEFT JOIN \
           (SELECT DISTINCT id, lon, lat\
             FROM nodes) t_pos \
          ON t_shops.id=t_pos.id)"
cur.execute(QUERY)
result = cur.fetchall()
print "----Mobile shops position on map----"
for i in result:
    print "Shop name: %s, Lat: %s, Lon: %s" %(i[0], i[1], i[2])
```

In [ ]:

```
----Mobile shops position on map----
Shop name: Telekom, Lat: 26.1652305, Lon: 44.4131738
Shop name: Vodafone, Lat: 26.1652997, Lon: 44.4131481
Shop name: Telekom, Lat: 26.0378174, Lon: 44.4216666
Shop name: Telekom, Lat: 26.1322591, Lon: 44.4610333
Shop name: Telekom, Lat: 26.1634361, Lon: 44.4428584
Shop name: Orange, Lat: 26.1009549, Lon: 44.4365981
Shop name: Vodafone, Lat: 26.1032041, Lon: 44.4319379
Shop name: Telekom, Lat: 26.155912, Lon: 44.44323
Shop name: Orange, Lat: 26.1567428, Lon: 44.4433002
Shop name: Vodafone, Lat: 26.155695, Lon: 44.4432368
Shop name: Telekom, Lat: 26.0512571, Lon: 44.4537926
Shop name: Orange, Lat: 26.0488647, Lon: 44.4535129
Shop name: Vodafone, Lat: 26.0493384, Lon: 44.4535546
Shop name: Vodafone, Lat: 26.0496848, Lon: 44.4531384
Shop name: Orange, Lat: 26.1649138, Lon: 44.4136472
Shop name: Vodafone, Lat: 26.0206955, Lon: 44.4170815
Shop name: Vodafone, Lat: 26.0207805, Lon: 44.4218188
Shop name: Telekom, Lat: 26.0209762, Lon: 44.4220013
Shop name: Orange, Lat: 26.0210729, Lon: 44.4220644
Shop name: Vodafone, Lat: 26.0999305, Lon: 44.4409042
Shop name: Telekom, Lat: 26.1048548, Lon: 44.4293456
Shop name: Vodafone, Lat: 26.1046681, Lon: 44.4292563
```

```
Shop name: Vodafone, Lat: 26.0481065, Lon: 44.4156653
Shop name: Orange, Lat: 26.0482113, Lon: 44.4156736
Shop name: Telekom, Lat: 26.0198371, Lon: 44.418722
Shop name: Vodafone, Lat: 26.0542288, Lon: 44.4356441
Shop name: Vodafone, Lat: 26.1652092, Lon: 44.4250328
Shop name: Vodafone, Lat: 26.1617167, Lon: 44.4245719
Shop name: Telekom, Lat: 26.0961319, Lon: 44.4578201
Shop name: Telekom, Lat: 26.06199, Lon: 44.4052198
Shop name: Orange, Lat: 26.0616935, Lon: 44.4050973
Shop name: Vodafone, Lat: 26.0613739, Lon: 44.4049717
Shop name: Orange, Lat: 26.1259489, Lon: 44.450451
Shop name: Orange, Lat: 26.0875623, Lon: 44.4512528
Shop name: Orange, Lat: 26.0878729, Lon: 44.4511025
Shop name: Orange, Lat: 26.1610099, Lon: 44.4244876
Shop name: Telekom, Lat: 26.1622579, Lon: 44.4246413
Shop name: Orange, Lat: 26.0975256, Lon: 44.4470335
Shop name: Vodafone, Lat: 26.0998087, Lon: 44.4527781
Shop name: Orange, Lat: 26.0676982, Lon: 44.4650354
Shop name: Telekom, Lat: 26.0922397, Lon: 44.3902586
Shop name: Orange, Lat: 26.092288, Lon: 44.3903602
Shop name: Orange, Lat: 26.1345424, Lon: 44.4408318
Shop name: Telekom, Lat: 26.0973459, Lon: 44.4452918
Shop name: Telekom, Lat: 26.0729787, Lon: 44.5511571
Shop name: Vodafone, Lat: 26.1220794, Lon: 44.394786
Shop name: Telekom, Lat: 26.1220311, Lon: 44.394437
Shop name: Orange, Lat: 26.1220472, Lon: 44.3945291
Shop name: Telekom, Lat: 26.1163739, Lon: 44.3864692
Shop name: Vodafone, Lat: 26.1243537, Lon: 44.448457
Shop name: Orange, Lat: 26.0473795, Lon: 44.4533448
Shop name: Vodafone, Lat: 26.1357505, Lon: 44.4160993
Shop name: Telekom, Lat: 26.15349, Lon: 44.3705855
Shop name: Vodafone, Lat: 26.0872516, Lon: 44.4652506
Shop name: Telekom, Lat: 26.1354927, Lon: 44.4394813
Shop name: Orange, Lat: 26.1338243, Lon: 44.3810137
Shop name: Vodafone, Lat: 26.1021383, Lon: 44.4793718
Shop name: Vodafone, Lat: 26.1040547, Lon: 44.4779982
```

Min and Max date by shop when the shop was included in openstreetmap

In [ ]:

```
QUERY = "SELECT value, MIN(SUBSTR(timestamp,1,10)),
MAX(SUBSTR(timestamp,1,10))FROM (\
        (SELECT id, value  \
            FROM nodes_tags WHERE value IN ('Telekom','Orange','Vodafone'))
t_shops \
          LEFT JOIN \
           (SELECT distinct id, timestamp\
             FROM nodes) t_date \
          ON t_shops.id=t_date.id) fin_table \
         GROUP BY fin_table.value  "
cur.execute(QUERY)
result = cur.fetchall()

for i in result:
    print "Shop name: %s, Min date: %s, Max date: %s" %(i[0], i[1], i[2])
```

In [ ]:

```
Shop name: Orange, Min date: 2012-05-11, Max date: 2017-02-10
Shop name: Telekom, Min date: 2012-03-29, Max date: 2016-11-27
Shop name: Vodafone, Min date: 2012-04-25, Max date: 2017-05-20
```

40% of the users contributed only once, while 67% of the contributions were made by the top 3 users. The most active user was razor74 with 33% of the total posts. One idea to increase the number of contributions per users, would be to emphasise the utility of the bots. A bot is "an automated or semi-automated tool that carries out repetitive and mundane tasks to help maintain OpenStreetMap" .

The number of mobile shops on the map is very low: Orange - 18, Vodafone - 21 and Telekom -19.

During xml parsing, I noticed that few Telekom shops were named Cosmote, even though the name changed in 2014 from Cosmote to Telekom. Maybe once the locations on the map are named and classified, the users don't bother to recheck the information and to update it if necessary. The last time information about a Telekom shop was added/changed on the map was an year ago, which in my opinion a a very long period of time.

Maybe the rewards (badges) for fixing an address is lower than creating one, so users are not motivated to update the existing information.

The typo errors I found and cleaned in the xml could be avoided by building a parser which parse the words input by the users.

Other ideas about the dataset

1 - There is a list of common street abbreviations for various languages on the openstreetmap wiki page: http://wiki.openstreetmap.org/wiki/Name_finder:Abbreviations#Rom.C3.A2n.C4.83_-_Romanian One of the problem I encounter in the project was related to the unabbreviated street names. I find reasonable that street names to be always in their unabbreviated form.

My suggestion is to add an auto formatting code which replace the abbreviated words with unabbreviated versions when uploading the files.

For example, if I type Str Preciziei, the script should replace the abbreviation after a space is typed or as soon as it is recognized. The final street name should be Strada Preciziei.

Pro: Using unabbreviated names would improve the searching of the streets.

Cons: This is a tedious job, because each country uses different abbreviations. And an auto-correct option can also easily lead to mistakes.

2 - The missing objectives on the map seems pretty difficult to spot. If addresses are missing in the OSM database then they will not show up in OSM file. In my project, I noticed that the list of mobile shops is incomplete.

Pro: Adding a special tag for missing data like “unsigned” or “none” would help contributors to find the missing data faster and to fill in the objective details. Helping contributors to easily find missing information on the map will improve the quality and completeness of the OpenStreetMap.

Cons: The name of this tag should be carefully chosen, so it won’t be misleading for some people. Must be sure that there isn’t an existing feature with the same name as the tag for missing objectives.

3 - Orange Romania has recently launched a prepay campaign which encourage clients to collect prizes and MB from certain locations on the map: https://www.orange.ro/4g-megahunt/. This kind of applications are familiarizing people with the geolocation and mapping concepts and can be used by

OpenStreetMap.

Pro: Fun way to gather map information.

Cons: It’s a private project and I’m guessing that the number of people who have an orange prepaid and are also willing to use the 4g-megahunt application is very small.

But there are other more popular games or applications which can be used as sources for the OSM: PokemonGo or the kind of apps we use to plan routes when jogging.

Conclusion

As expected, data inserted by humans is not perfect, it's almost certain to contain mistakes. The OpenStreetMap data of Bucharest is obviously incomplete and contains old information. For this kind of project, you need active users contributing to the map area and a good reward system will keep them motivated to constantly update the map.