OpenStreetMap Sample Project

To analyze OSM data I have chosen an area that I am familiar with: Munich, the city I live in.

Map Area: Munich, Germany

http://overpass-api.de/api/map?bbox=11.4100,48.0701,11.7293,48.2080

1. Lession 6 questions

The code and the results from running it can be found in the file “Data Analyst ND P3.html”

1.1 Count the tags

Counting the tags in the OSM extract:

```
{'node': 1486269, 'member': 137545, 'remark': 1, 'nd': 1959807, 'tag':
1634060, 'note': 1, 'meta': 1, 'relation': 5421, 'way': 291987, 'osm': 1}
```

1.2 Count the key types

Counting the key types in the OSM extract:

```
{'lower': 802489, 'lower_colon': 821809, 'other': 9424, 'problemchars':
338}
```

I further analyzed the problematic key types and found out that the only problem is that “.” is used in the name. List of key types with problematic characters:

```
set(['addr.source:housenumber',
     'contact.source:website',
     'footway:left.sloped_curb.end',
     'footway:left.sloped_curb.start',
     'footway:left.smoothness',
     'footway:left.surface',
     'footway:left.wheelchair',
     'footway:left.wheelchair.end',
     'footway:left.wheelchair.start',
     'footway:left.width',
     'footway:right.sloped_curb.end',
     'footway:right.sloped_curb.start',
     'footway:right.smoothness',
     'footway:right.surface',
     'footway:right.wheelchair',
     'footway:right.wheelchair.end',
     'footway:right.wheelchair.start',
     'footway:right.width',
     'step.condition',
     'step.height',
     'surface.material'])
```

1.3 Get Users

The number of different users found in the OSM extract is quite large. They will be counted with MongoDB later on.

1.4. Audit street names

The street names in Germany are not so easy to categorize. There are many more endings possible that just “street” or “place” (in German: “Straße” and “Platz”). I added a larger amount of possible endings that could be the last word in the street name (like “Pullacher Straße”) or the ending (like “Lutherstraße” or “Martin-Luther-Straße”).

```
expected = [u'Stra\xdfe', "Allee", "Platz", "Weg", "Ring",
            u"Chause\xe8", u'Br\xfccke', "Berg", "Park",
            "Feld", "Graben", "Holz", "Siedlung", "Steig", "Weiher",
            "Bach", "Anger"]

expectedEnds = ["weg", "-Weg","-Platz", "platz", u'stra\xdfe',
                u'-Stra\xdfe', "promenade", "-Bogen", "-Ring",
                "gasse", "-Gasse", "allee", "-Allee", "bogen", "graben",
                "see", "feld", "leiten", "berg", "ring",
                "rain", "burg", "leite", "bach", "tal", "wiese",
                "garten", "anger", "acker", "thal", "hof", "wald",
                "-Anger", "ried", "dorf", "-Damm", "damm", "-Passage",
                u"m\xfchle", "gassl", "steig", u'-Br\xfccke',
                "park", "pfad", "wies", "viertel", "ufer", "hof", "haus",
                "weide", "kanal", "-Hof", "mark", "winkel",
                "zentrum", "hausen", "bauer", "blick", "hofen", "wall",
                "brunnen", "breite", "hain", "fleck",
                "passage", "schlag", "tor", "holz", "hort", "-Siedlung",
                "weiden", "stein", "kam", "grube", "grund"]
```

The resulting list of street names was still long but showed no abbreviations or other abnormalities.

1.5 Convert XML to JSON

The conversion of XML to JSON went smooth. The key types with “.” in the key were converted to so that “.” was replaced with “:”, because “.” is not allowed in MongoDB as a key value.

2. Problems encountered

2.1 Runtimes

The runtimes of the XML parsing was long. To check if there is any progress I added the printout of a star (“*”) every 300.000 nodes processed. This way I was able to verify that the program is making progress.

2.2 Invalid key names

Some key names used in the XML extract are not compatible with the allowed key values of MongoDB. Instead of leaving them out I adjusted them slightly so that they are included in the JSON and could be loaded. See 1.5 for details.

3. Data Overview

3.1 File sizes

```
MUC.osm:         369 MB
MUC.osm.json:    390 MB
```

3.2 Number of documents

```
> db.MUC.find().count()
1778256
```

3.3 Number of nodes

```
> db.MUC.find({"type":"node"}).count()
1484415
```

3.4 Number of ways

```
> db.MUC.find({"type":"way"}).count()
291892
```

3.5 Number of unique users

```
> db.MUC.distinct("created.user").length
2494
```

3.6 Top 1 contributing user

```
> db.MUC.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":
1}}},{"$sort":{"count":-1}},{"$limit":1}])
{ "_id" : "rolandg", "count" : 205382 }
```

3.7 Top 10 cuisines

```
> db.MUC.aggregate([{"$match":{"cuisine":{"$exists":
1},"amenity":"restaurant"}},{"$group":{_id:"$cuisine", sum:{$sum:1}}},
{"$sort":{sum:-1}},{"$limit":10}])
{ "_id" : "italian", "sum" : 335 }
{ "_id" : "regional", "sum" : 119 }
{ "_id" : "greek", "sum" : 84 }
{ "_id" : "bavarian", "sum" : 82 }
{ "_id" : "indian", "sum" : 54 }
{ "_id" : "asian", "sum" : 51 }
{ "_id" : "vietnamese", "sum" : 48 }
{ "_id" : "german", "sum" : 47 }
{ "_id" : "international", "sum" : 45 }
{ "_id" : "chinese", "sum" : 44 }
```

3.8 Top 10 amenities

```
> db.MUC.aggregate([{"$match":{"amenity":{"$exists":1}}},{"$group":
{_id:"$amenity", Anz:{$sum:1}}},{"$sort":{Anz:-1}},{"$limit":10}])
{ "_id" : "parking", "Anz" : 3644 }
{ "_id" : "bench", "Anz" : 2986 }
{ "_id" : "restaurant", "Anz" : 1768 }
{ "_id" : "vending_machine", "Anz" : 1273 }
{ "_id" : "recycling", "Anz" : 1012 }
{ "_id" : "bicycle_parking", "Anz" : 919 }
{ "_id" : "shelter", "Anz" : 840 }
{ "_id" : "post_box", "Anz" : 772 }
{ "_id" : "telephone", "Anz" : 653 }
{ "_id" : "cafe", "Anz" : 638 }
```

3.9 Checking an address

Checking an address of a church I know gives the correct result:

```
> db.MUC.find({"name":"Lutherkirche"},{"_id":0,"building":1,"name":
1,"address":1})
{ "building" : "church", "name" : "Lutherkirche", "address" :
{ "housenumber" : "4", "postcode" : "81539", "country" : "DE", "city" :
"München", "street" : "Martin-Luther-Straße" } }
```

4. Additional Ideas

4.1 Cuisine inconsistencies

If you look at the values for “cuisine” that contain “italian” you find some lists that should show that the cuisine is italian and more.

```
> db.MUC.distinct("cuisine", {"cuisine":/italian/,
"amenity":"restaurant"})
[
      "italian",
      "italian;asian",
      "italian;pizza",
      "italian;mexican",
      "italian;regional;german;coffee_shop",
      "italian;turkish",
      "italian;pizza;pasta",
      "indian;italian"
]
```

So looking at the top 10 cuisines like in in 3.7 is not 100% correct. A way to process the lists in the cuisine value must be included. The number of italian restaurants is slightly higher than shown in 3.7:

```
> db.MUC.find({"cuisine":/italian/, "amenity":"restaurant"}).count()
344
```

Also, “pizza” as an value itself might count as an italian restaurant.

```
> db.MUC.distinct("cuisine", {"cuisine":/pizza/,"amenity":"restaurant"})
[
      "pizza",
      "greek;pizza",
      "italian;pizza",
      "ice_cream,pizza",
      "italian;pizza;pasta",
      "kebab,pizza"
]
```

This shows that the values of “cuisine” might be used inconsistently and applications and users looking for specific cuisines have to try several variants to find all restaurants.

4.2 Religion inconsistencies

Looking at religious places showed a large number for the christian religion.

```
> db.MUC.aggregate([{"$match":{"amenity":"place_of_worship","religion":
{"$exists":1}}},{"$group":{"_id":"$religion","count":{"$sum":1}}},
{"$sort":{"count":-1}}])
{ "_id" : "christian", "count" : 345 }
{ "_id" : "muslim", "count" : 12 }
{ "_id" : "buddhist", "count" : 4 }
{ "_id" : "jewish", "count" : 3 }
{ "_id" : "hindu", "count" : 1 }
{ "_id" : "sikh", "count" : 1 }
{ "_id" : "scientologist", "count" : 1 }
```

That might be correct for Munich but the high number should be checked in more detail. So I analyzed the denomination:

```
> db.MUC.aggregate([{"$match":{"religion":"christian","denomination":
{"$exists":1}}},{"$group":{"_id":"$denomination","count":{"$sum":1}}},
{"$sort":{"count":-1}}])
```

```
{ "_id" : "catholic", "count" : 198 }
{ "_id" : "protestant", "count" : 60 }
{ "_id" : "roman_catholic", "count" : 36 }
{ "_id" : "jehovahs_witness", "count" : 7 }
{ "_id" : "new_apostolic", "count" : 7 }
{ "_id" : "lutheran", "count" : 6 }
{ "_id" : "romanian_orthodox", "count" : 5 }
{ "_id" : "russian_orthodox", "count" : 3 }
{ "_id" : "orthodox", "count" : 2 }
{ "_id" : "pentecostal", "count" : 2 }
{ "_id" : "mormon", "count" : 2 }
{ "_id" : "free_evangelical", "count" : 2 }
{ "_id" : "methodist", "count" : 2 }
{ "_id" : "old_catholic", "count" : 1 }
{ "_id" : "scientist", "count" : 1 }
{ "_id" : "serbian_orthodox", "count" : 1 }
{ "_id" : "ethiopian_orthodox", "count" : 1 }
{ "_id" : "greek_orthodox", "count" : 1 }
{ "_id" : "christian_community", "count" : 1 }
{ "_id" : "bulgarian_orthodox", "count" : 1 }
```

The number for the denomination “catholic” is still very high but reducing it to church buildings gave a reasonable result:

```
> db.MUC.aggregate([{"$match":
{"religion":"christian","building":"church","religion":{"$exists":1}}},
{"$group":{"_id":"$denomination","count":{"$sum":1}}},{"$sort":
{"count":-1}}])
{ "_id" : "catholic", "count" : 52 }
{ "_id" : "protestant", "count" : 50 }
{ "_id" : "roman_catholic", "count" : 13 }
{ "_id" : null, "count" : 6 }
{ "_id" : "lutheran", "count" : 2 }
{ "_id" : "romanian_orthodox", "count" : 1 }
{ "_id" : "orthodox", "count" : 1 }
{ "_id" : "serbian_orthodox", "count" : 1 }
{ "_id" : "anglican", "count" : 1 }
{ "_id" : "scientist", "count" : 1 }
{ "_id" : "russian_orthodox", "count" : 1 }
{ "_id" : "mormon", "count" : 1 }
{ "_id" : "old_catholic", "count" : 1 }
```

What you see here is that there are denominations that are not different but have different names. “catholic” and “roman_catholic” are the same. Also the difference between “lutheran” and “protestant” is not clear as these not exist in reality. Free protestant churches are missing totally and some churches have no denomination at all.

The meta data of the churches should be cleaned in a more consistent way.