Data Wrangling with OpenStreetMap and MongoDB

Grace Pehl, PhD

Map Area: Florida's Treasure Coast region (lat. 27 to 28, long. -81 to -80)

http://www.openstreetmap.org/export#map=9/27.5022/-80.5009

Problems Encountered in the Map

- Over-abbreviated street types were expanded using a mapping.

```
mapping = { "St": "Street", "St.": "Street", "Ave": "Avenue", "ave": "Avenue",
            "Rd.": "Road", "Pl": "Place", "Ct": "Court", "Dr.": "Drive",
            "Dr": "Drive", "Blvd": "Boulevard", "BLVD": "Boulevard",
            "SE": "Southeast"}
```

- State appeared as both “FL” and “Florida.” Since the US postal system uses state abbreviations, “FL” was used as standard.

```
if key == "addr:state" & value == "Florida":
    value = "FL"
```

- The city of Hobe Sound was listed as Hobe Sound, FL in the city field.

```
if key == "addr:city" & value == "Hobe Sound, FL":
    value = "Hobe Sound"
```

- Despite the region's many faiths, only three religions are present in data: christian, jewish, and unitarian_universalist. This indicates that the map needs additional user input.

Overview of the Data

Statistics of the OSM file:

OSM file size: 82,586 kB
JSON file size: 89,959 kB
Tags:

'member'	24700
'meta'	1
'nd'	446341
'node'	373451
'note'	1
'osm'	1
'relation'	373,
'tag'	229216
'way'	33166

Unique users: 273
Type of keys: 621 unique keys used
'lower': 85955, 'lower_colon': 133910, 'other': 9351, 'problemchars': 0

MongoDB Queries

Sample document: *db.osm.find_one()*

```
_id': ObjectId('55b32875d44d5c1768b3d426'),
 u'created': {u'changeset': u'14920451',
              u'timestamp': u'2013-02-05T10:53:40Z',
```

u'uid': u'207745',
u'user': u'NE2',
u'version': u'3'},
u'id': u'26786875',
u'pos': [27.6932111, -80.8890663],
u'type': u'node'}

Total number of documents: 406,617 db.osm.find().count()
Number of nodes: 373,450 db.osm.find({"type":"node"}).count()
Number of ways: 33,166 db.osm.find({"type":"way"}).count()

Number of unique users (by user id)**:** 261

```
pipeline = "[{"$group":{"_id":"$created.uid", "count":{"$sum":1}}}]"
db.osm.aggregate(pipeline)
```

Top 5 contributing users (by user name)**:**

user	**contributions**
"grouper"	157,846
"woodpeck_fixbot	53,623
"NE2"	52,685
"Latze"	14,854
"Chris Lawrence"	12,198

```
pipeline = "[{"$group" :{ "_id" : "$created.user",
                          "count" : {"$sum" : 1 }}},
             { "$sort" : { "count" : -1 }},
             { "$limit" : 5}]"
db.osm.aggregate(pipeline)
```

Number of users contributing 1 entry (by user name)**:** 46

```
pipeline = "[{"$group":{"_id":"$created.uid", "count":{"$sum":1}}},
             {"$group":{"_id":"$count", "num_users":{"$sum":1}}},
             {"$sort":{"_id":1}},
             {"$limit":1}]"
db.osm.aggregate(pipeline)
```

Number of amenities: 1065 db.osm.find({"amenity":{"$exists": 1}}).count()

Other Ideas about the Dataset

Key prefixes

In the dataset, 621 different keys were used to describe the data. Listing them showed many keys carrieds a prefix, often "tiger:" or "gnis:" A search revealed that tiger is an acronym used for a spatial extract from the US Census Bureau and gnis stands for geographic names information system used by the US Geological Survey. A further cleaning step could be to remove these prefixes from the key and create another key = "source" with value = "tiger" or "gnis".

"Name:" keys

There are also hundreds of keys that seem useless, called "name:" followed by 2-3 random letters such as "bcl", "rw", "kv", "diq", or "tpi". These keys could be investigated and possibly removed from the dataset.

Improvements to OSM

Data analysis using Open Street Maps is complicated by the non-standardized formats. It would be helpful if users had more guidance as they entered data. Drop-down boxes would be useful for the most commonly entered data like address and amenity, but would become to cumbersome if used for all types of user-entered data. Perhaps the 10 most common keys could be in a drop-down box, with a write-in "other" choice available. A few short style guidelines ("Avoid abbreviations except for countries and states.") or samples of good input would guide users to entering cleaner data, without giving up the flexibility of the OSM project.

Additional data exploration using MongoDB queries

Number of amenities: 1065 `db.osm.find({"amenity":{"$exists": 1}}).count()`

Types of amenities:

```
pipeline = "[{"$group": {"_id": "$amenity", "count":{"$sum":1}}},
                {"$sort" : {"count" : -1}}]"
db.osm.aggregate(pipeline)
```

{u'count': 321, u'_id': u'place_of_worship'}
{u'count': 159, u'_id': u'parking'}
{u'count': 148, u'_id': u'school'}
{u'count': 67, u'_id': u'restaurant'}
{u'count': 62, u'_id': u'fuel'}
{u'count': 58, u'_id': u'fire_station'}
{u'count': 50, u'_id': u'fast_food'}
{u'count': 24, u'_id': u'bank'}
{u'count': 20, u'_id': u'library'}
{u'count': 19, u'_id': u'pharmacy'}
{u'count': 17, u'_id': u'police'}
{u'count': 16, u'_id': u'post_office'}
{u'count': 14, u'_id': u'hospital'}
{u'count': 12, u'_id': u'toilets'}
{u'count': 12, u'_id': u'fountain'}
{u'count': 10, u'_id': u'grave_yard'}
{u'count': 8, u'_id': u'cafe'}
{u'count': 7, u'_id': u'swimming_pool'}
{u'count': 6, u'_id': u'atm'}
{u'count': 4, u'_id': u'community_centre'}
{u'count': 4, u'_id': u'theatre'}
{u'count': 3, u'_id': u'car_wash'}
{u'count': 3, u'_id': u'prison'}
{u'count': 3, u'_id': u'public_building'}
{u'count': 2, u'_id': u'auto:service'}
{u'count': 2, u'_id': u'parking_aisle'}
{u'count': 2, u'_id': u'dentist'}
{u'count': 2, u'_id': u'shelter'}
{u'count': 2, u'_id': u'bar'}
{u'count': 1, u'_id': u'college'}
{u'count': 1, u'_id': u'boat_storage'}
{u'count': 1, u'_id': u'university'}
{u'count': 1, u'_id': u'department_store'}
{u'count': 1, u'_id': u'animal_shelter'}
{u'count': 1, u'_id': u'doctors'}
{u'count': 1, u'_id': u'townhall'}
{u'count': 1, u'_id': u'social_centre'}