OpenStreetMap Data Wrangling

Michael duPont - Orange County, FL

http://overpass-api.de/api/map?bbox=-81.6552,28.3491,-81.1299,28.6117

	28.6117	
-81.6552	Orange County (Orlando)	-81.1299
	28.3491	

This bounding box fits much of Orange County, FL. It contains all of Orlando and most of its suburbs, tourist areas like Walt Disney World and Universal Studios, airports, industrial areas, state parks, and the University of Central Florida.

Sections

1. Problems Encountered in the Map

Possible Key Collisions

After performing an initial key audit, I found some possible key collisions that would have tripped up a JSON port. An example of this is the keys “building”, “building:height”, and “building:levels”. The port would want to create a value of type string and then attempt to treat is as a dictionary. To prevent this from happening, the code would recognize that the existing value is not a dictionary, and create a redundant key pair in the resulting dictionary. Our example would yield the following:

```
'building' : {
    'building' : val,
    'height' : val,
    'levels' : val
}
```

Address Lengthening and Standardization

During the initial audit, I started the eventual cleaning process by analyzing an item's address tags. I started by recognizing all street name abbreviations and expanding them. This would include expanding "Cir" to "Circle". There were no direction elements in the street name as they had their own tag and were already expanded.

Next I made sure that every county, state, and country value equalled "Orange", "Florida", and "US" respectively because the bounding box was limited to only Orange County. This turned out to be true, so, when creating the JSON port, any item with an address field would have those three fields added if they weren't already present.

Another field that required fixing was city name. These values had no standard capitalization. While most cities looked like "Orlando" and "Windermere", some values were all uppercase like "LONGWOOD", all lowercase like "gotha", or mixed/mistyped like "Winter pArk". This was fixed by only having the first letter of each word capitalized since no city name in the set contained articles.

Last was the postcode. Only a few postcodes included a state prefix ("FL 32803") or were Zip+4 codes ("32803-4267"), so these were both stripped.

This cleaning method is demonstrated by the following example address field's "before" on the left and "after" on the right:

```
'address' : {
    'city' : 'orlando',
    'direction' : 'East',
    'housenumber' : '4532',
    'postcode' : 'FL 32803-8472',
    'street' : 'Colonial Dr.'
}
```

```
'address' : {
    'city' : 'Orlando',
    'country' : 'US',
    'county' : 'Orange',
    'direction' : 'East',
    'housenumber' : '4532',
    'postcode' : '32803',
    'state' : 'Florida',
    'street' : 'Colonial Drive'
}
```

Phone Number Standardization

The phone numbers were entered in a wide variety of formats. I chose to go with the option that I feel is both easy to read and easy to parse: "+1 407-859-3954". This is achieved by stripping out the original country code and any non-alphanumeric characters and then inserting the remaining values into the template. It will also work with "1-800" phone numbers.

Typos and Further Standardization

The cleaning process would also perform a number of targeted corrections on a key-by-key basis. This could include setting values to lowercase, replacing underscores with spaces, splitting values by semicolon into lists, and replacing misspelled key strings with their proper or

standardized values. Doing this for fields such as “name” or “operator” would make them easier to query later.

2. Data Overview

File Size

orlandomap.osm - 170.6 MB
orlandomap.json - 169.5 MB

Number of Documents

```
> db.orangecounty.find().count()
825021
```

Number of Nodes

```
> db.orangecounty.find({'type':'node'}).count()
729423
```

Number of Ways

```
> db.orangecounty.find({'type':'way'}).count()
95598
```

Number of Unique Contributing Users

```
> db.orangecounty.distinct('created.user').length
439
```

Most Recent Update

```
> db.orangecounty.aggregate([
       {'$sort':{'created.timestamp':-1}},
       {'$limit':1},
       {'$project':{'_id':'$created.timestamp'}} ])
{ "_id" : "2015-08-03T01:10:09Z" }
```

The osm file used was downloaded on 2015-08-05, three days after the data was last updated.

Earliest Update

```
> db.orangecounty.aggregate([
       {'$sort':{'created.timestamp':1}},
       {'$limit':1},
       {'$project':{'_id':'$created.timestamp'}} ])
{ "_id" : "2007-09-28T11:26:19Z" }
```

This earliest update matches the 2007 import of the TIGER road census data, effectively adding the US to the Open Street Maps database.

3. Additional Queries

Cross-Referencing for Additional Information

One way to improve upon this dataset would be to cross-reference information from additional data sources. For example, we can use the free web API GeoNames to add information like neighborhood and elevation to elements that contain a valid “pos” attribute. Sample code for retrieving neighborhood name is show below.

```
import json
from requests import requests
dataset = jsonload('myosmdata.json')
url = 'http://api.geonames.org/neighbourhood?lat={0}&lng={1}&username={2}'
uid = 'myAuthToken'
for element in dataset:
    if 'pos' in element:
        geoResponse = requests.get(url.format(element['pos']['lat'] , element['pos']['lon'] ,
uid))
        try: element['neighborhood'] = geoResponse['geonames']['neightborhood']['name']
        except: pass
json.dump('myosmdata.json')
```

Most Frequent Religious Denominations

```
> db.orangecounty.aggregate([
       {'$match':{'denomination':{'$exists':1}}},
       {'$group':{'_id':'$denomination','count':{'$sum':1}}},
       {'$sort':{'count':-1}},
       {'$limit':5}])
{ "_id" : "baptist", "count" : 77 }
{ "_id" : "methodist", "count" : 25 }
{ "_id" : "presbyterian", "count" : 20 }
{ "_id" : "catholic", "count" : 13 }
{ "_id" : "lutheran", "count" : 10 }
```

Items Operated by Disney

```
> db.orangecounty.find({'operator':{'$regex':'.*Disney.*'},'type':'node'}).count()
55
> db.orangecounty.find({'operator':{'$regex':'.*Disney.*'},'type':'way'}).count()
194
```

Western-Most Tourist Attraction

```
> db.orangecounty.aggregate([
       {'$match':{'tourism':{'$exists':1},'tourism':'attraction','pos':{'$exists':1}}},
       {'$sort':{'pos.lon':-1}},
       {'$limit':1},
       {'$project':{'_id':'$name','pos':'$pos'}}]).pretty()
{
```

```
    "_id" : "Nile Crocodiles",
    "pos" : {
        "lat" : "28.3614615",
        "lon" : "-81.5951715"
    }
}
```

This is the “Nile Crocodiles” exhibit/pen at Disney’s Animal Kingdom