OpenStreetMap Project – Data Wrangling with MongoDB

Maxwell Edwards – 11/13/15
Syracuse, NY, United States
Data available at: https://s3.amazonaws.com/metro-extracts.mapzen.com/syracuse_new-york.osm.bz2

1. Problems encountered in your map

<u>Issues with 'addr:' key</u>

Parsing keys with "addr:" was receiving the following error when I ran the 'data.py': TypeError: 'str' object does not support item assignment. This took me some time to figure out why it was happening. I figured out that I needed to examine the 'addr:' key values more thoroughly. I did this by adding a try / except statement and printing out the nodes where the error was occurring. I found that 'address' contained strings that I was not expecting. Therefore, I added an if statement to remove 'addr:interpolation', addr:full, 'addr:housename'.

As mentioned, this took me some time to figure out. The first thing I tried was checking that the node dictionary was not being assigned an 'address' key prior to parsing the <tags>. I wrote a simple script (see below) to check these which generated a set of all the attributes to make sure there wasn't an 'address' attribute seeking in.

```
#CODE TO CHECK ELEMENT ATTRIBUTES
def check_elements(file_in):

   attrib_set = set()
    for _, element in ET.iterparse(file_in):
      for a in element.attrib:
          attrib_set.add(a)

    return attrib_set
```

```
#OUTPUT
set(['changeset', 'generator', 'id', 'k', 'lat', 'lon', 'maxlat',
    'maxlon', 'minlat', 'minlon', 'ref', 'role', 'timestamp', 'type',
    'uid', 'user', 'v', 'version'])
```

<u>Several Unwanted Tags</u>

When I used data.py to generate my first JSON document and loaded it into MongoDB, I quickly realized there were several unwanted tags. I remove the tags using the function below passing the tag.attrib['k'] value as a parameter.

```
IGNORE = ["ele"]
IGNORE_PRE = ["gnis:", "is_in", "name:", "tiger", "sourc"]

def ignore_keys(key_value):
    if key_value in IGNORE:
        return True
    elif key_value[:5] in IGNORE_PRE:
        return True
    return False
```

Street Types

I used the audit.py file to audit street types similary to how we did for the Chicago.osm file. For Syracuse, the audit did not find any issues. There were several street types that weren't in my expected list but they were valid (i.e. Terrace, Circle). Surprisingly, there were no street types that were abbreviated. Maybe someone already cleaned this file?

Zip Code not in Syracuse, NY

I audited the 'addr:postalcode' similarly to street type in Lesson 6 exercises. Initially I found several zip codes that weren't Syracuse zip codes but after investing each one, I only found two zip codes that make no sense (i.e. not a neighborhood town/city of Syracuse). They were set([None, '13507', '14224']). '13507' isn't a zip code anywhere so my guess is that was a typo given its starts with '13' (all Syracuse zip codes start with '13'). '14224' is Buffalo, NY which about a 3 hour drive away. To clean these, I added code in data.py to remove the zip values for elements that contain these postal codes. Additionally, I added code to strip any characters after the first 5 so nine digit zip codes were removed or zip codes that contained a "-" after the first 5 characters.

Prior to clean	**After Clean**

db.project.find({"address.postcode" : "13507"}).count()
4
db.project.find({"address.postcode" : "14224"}).count()
68
	db.project.find({"address.postcode" : "13507"}).count()
	0
	db.project.find({"address.postcode" : "14224"}).count()
	0

Incorrect city values

I audited 'addr:city' and found some issues. For some of the city values, a state value was part of the string (i.e. East Syracuse, NY). Other cities were simply misspelled (i.e. Mattdale should be Mattydale). These were cleaned in data.py accordingly based on what I found in this audit by mapping the "bad" values to the good values. I ran queries in MongoDB to confirm these values were cleaned correctly. For example, my "Mattydale" count went up by one after I cleaned "Mattdale" – meaning there was 1 incorrect spelling of this city in the XML file.

2. Overview of the Data

The size of the JSON and OSM files are as follows:

```
11/13/2015  05:41 PM        67,956,679 syracuse_new-york.json
11/11/2015  10:36 AM        63,407,569 syracuse_new-york.osm
```

Total number of documents

```
>>> db.project.find().count()
1230720
```

Total number of nodes

```
>>> db.project.find({"type" : "node"} ).count()
1092436
```

Total number of ways

```
>>> db.project.find({"type" : "way"} ).count()
138208
```

Total number of ways

```
>>> len(db.project.distinct("created.user"))
212
```

Top 3 users and count of their contributions

What is interesting is the top 3 contributions account for 83.5% of the total contributions.
Top 3 Total contributions = 1027428 (sum of output below)
Total Contributions

Input (using aggregate(pipeline))	**Output**
pipeline.append({ "$match" :	[{u'_id': u'zeromap', u'count': 618016},
{ "created.user" :	{u'_id': u'woodpeck_fixbot', u'count': 303016},
{ "$exists" : 1 } } })	{u'_id': u'DTHG', u'count': 106396}]
pipeline.append({ "$group" :	
{ "_id" : "$created.user",	
"count" :	
{ "$sum" : 1 } } })	
pipeline.append({ "$sort" :	
{ "count" : -1 } })	
pipeline.append({ "$limit" : 3 })	

Total docs with an address field

Only 1.9% of our node/ways actually have an address defined

```
>>> db.project.find( { "address" : { "$exists" : 1 } } ).count()
23384
```

Top 5 amenities in dataset

It is interesting that parking is the #1 amenity, however these are mostly formed by way element tags. When I queried the parking addresses, only 4 actually had an address.

Input (using aggregate(pipeline))	**Output**
	[{u'_id': u'parking', u'count': 3636},
pipeline.append({ "$match" : {"amenity" : { "$exists":1 } } })	{u'_id': u'school', u'count': 760},
	{u'_id': u'bench', u'count': 576},
pipeline.append({ "$group" : { "_id" : "$amenity", "count" : { "$sum": 1 } } })	{u'_id': u'restaurant', u'count': 564},
pipeline.append({ "$sort" : {"count":-1} })	{u'_id': u'fast_food', u'count': 520}]
pipeline.append({ "$limit" : 5 })	

Top 5 restaurant cuisines in Syracuse

I decided to look into restaurants more and queried by cuisine type. No surprise to me that pizza is number one (other than ‘None’) as Syracuse has a rich Italian population.

Input (using aggregate(pipeline))	**Output**
pipeline.append({"$match": {"amenity": "restaurant" } }) pipeline.append({"$group": {"_id": "$cuisine", "count": {"$sum" : 1 } } }) pipeline.append({ "$sort" : { "count" : -1 } }) pipeline.append({ "$limit" : 5 })	[{u'_id': None, u'count': 244}, {u'_id': u'pizza', u'count': 60}, {u'_id': u'american', u'count': 40}, {u'_id': u'chinese', u'count': 32}, {u'_id': u'italian', u'count': 32}]

3. Other ideas about the datasets

“FIXME” and “fixme” tags

I wanted to know all the key fields that existed so I used the following function (see Reference 1 for where the code came from). Several of the key value pairs weren’t filtered out in data.py that could have been filtered out (I missed these). For example, there are several key fields labeled “FIXME” and “fixme” which essentially served as comment fields such as “verified via digital data that segment does not have a name please verify via ground survey” and “not in Bing”. To make the data cleaner, I would remove these fields similar to how I removed others as shown above.

```
def get_keys():

    mapper = Code("""
        function() {
                    for (var key in this) { emit(key, null); }
                }
    """)
    reducer = Code("""
        function(key, stuff) { return null; }
    """)

    ans = db.project.map_reduce(mapper, reducer
    , out = {'inline' : 1}
    , full_response = True)

    return ans
```

How to Improve the Quality of the Dataset

The Syracuse OpenStreetMap dataset obviously contained some issues (zip codes in Buffalo, incorrect spellings, etc). User participation is an issue as the majority of the inputs are completed by 3 users. One way to improve user participation would be to start a “meetup” group to improve upon the OpenStreetMap data in the city in which you live. The members could collect data in the city and update/add to the dataset, clean the dataset, and think about other ways to improve the dataset. This group would be beneficial for the members and for the OpenStreetMap movement. The members would be learning a lot of useful information such as how to work with large amounts of data (similar to what we learned in this class) and the map data would obviously improve. The main problem with implementing this meetup group would be member retention. Most people don’t have the time in their lives to constantly work on this for free.

References

1. http://stackoverflow.com/questions/2298870/mongodb-get-names-of-all-keys-in-collection
2. https://docs.mongodb.org/manual/
3. https://api.mongodb.org/python/current/
4. http://stackoverflow.com/questions/12451431/loading-and-parsing-a-json-file-in-python