Udacity DAND Open Street Maps Data Wrangling - Orlando

Map Area

Orlando Florida

Wrangling Challanges

Tag Sub Values

A deign decision of the OSM data is that the tags use a key value structures to encode the tag infortiom using a key value pair. While this is helpful for processing the XML data it isn't needed in a JSON/dictionary type structure.

```
<tag k="name" v="Duff Beer"/>
```

It was stripped away using the fix_key_value_pairs function. The dictionar is updated with a new key, value pair comprised of the k attribute v attribute.

```
elem['attribs'].update( { elem['attribs']['k'] :elem['attribs']['v']})
    del elem['attribs']['k']
    del elem['attribs']['v']
    print cur.replace_one({'_id': elem['_id']},  elem, False)
```

This would transfer the above tag into the following.

```
'name': 'Duff Beer'
```

This transformation needed to occur on the inital node level as well as the child tag sub level.

Fix Street Endings

The data had street endings that were non standard as shown below.

```
    <tag k="addr:street" v="Garden Parks Blvd."/>
    <tag k="addr:street" v="Universal Blvd"/>
    <tag k="addr:street" v="E Michigan St."/>
```

The follwing code uses a regex and replace function to replace the street endings with mappings in a dictionary. The regex pulls out the last string following a space in the addr:street tag.

```
split_endings_re_search = split_endings_re.search(street).group()
    if split_endings_re_search in street_mappings:
        #replace the incorrect entry identifyed with the dict mapping
        return re.sub(split_endings_re, street_mappings[split_endings_re_search], stree
```

Fix Zip Codes

The Zip codes are another area of the adress that has lots of issues and is realatively easy to clean. Prior to changin the Postal code there were a few node tags that included the state along with the zip code.

```
        <tag k="addr:postcode" v="FL 32792"/>
        <tag k="addr:postcode" v="FL 32803"/>
        <tag k="addr:postcode" v="FL 32803"/>
```

This was cleaned by utilizing the same regex expression used for the street name identification. The code first checks for zipcodes that aren't 5 or 10 characters long first. This reduces the amount of checks that needs to be made for the zip code. It then checks for any spaces in the zipcode and if it's found overwrites the postcode value with the string after the space. There is also a check for zipcodes that don't start with 3 indicatiing it's not in Florida.

```
if (len(zip) != 5) and (len(zip) !=10):
        print 'zip error to fix: '+ zip
        print len(zip)
        if zip.find(' ') > 0:
            print zip.find(' ')
            return split_endings_re.search(zip).group()

    elif int(zip[0]) != 3:
        print 'Error: Zip not in FL: '+ zip
        print split_endings_re.search(zip)
```

Overview and Data Stats

File Size

orlando_florida.osm 128M

Collection Stats

```
{'num_elem_school': 41,
 'num_high_school': 11,
 'num_publix': 39,
 'total_db_size': 23715840,
 'unique_nodes': 569564,
 'unique_users': 679,
 'unique_ways': 85145}

 def dbStats():
    stats = {
                'total_db_size': db.command('collStats', DBNAME)['storageSize'],
                'unique_users' : len(cur.distinct('userid')),
                'unique_nodes' : cur.count( {'type':'node'} ),
                'unique_ways' : cur.count( {'type':'way'} ),
                'num_publix' : cur.count({'name':re.compile('.*ublix.*')}),
                'num_high_school' : cur.count({'name': re.compile('.*High School.*')}),
                'num_elem_school' : cur.count({'name': re.compile('.*Elementary School.


    }

    pprint.pprint( stats)
```

Top Contributors to Map:

1. NE2 made 577754 contributions
2. 3yoda made 82202 contributions
3. crystalwalrein made 68620 contributions
4. epcotfan made 46372 contributions
5. Brian@Brea made 34778 contributions
6. RobChafer made 32460 contributions
7. dale_p made 30286 contributions
8. KindredCoda made 30050 contributions

9. Adam Martin made 29976 contributions
10. grouper made 27508 contributions

```
def topContributors():
    pipeline = [{'$match': {'attribs.user': {'$exists': True}}},
                {'$group': {'_id':'$attribs.user', 'count': {'$sum':1}}}, \
                {'$sort': {'count':-1}}, \
                {'$project':{ 'uid': 1, 'user':1, 'count':1}}, \
                {'$limit': 10} ]

    print 'Top Contributors to Map:'
    for i, user in enumerate(cur.aggregate(pipeline), start=1):
        print str(i)+'. '+user['_id']+' made '+str(user['count'])+' contributions'
```

Additional Ideas

GPS Location Convergence

A lot of the nodes seem to be revisions to the GPS coordinates. A sum of squared computation can reduce the distance down to a scalar metric that can be used to converge the GPS coordinates by rejecting GPS coordinates that differ too drastically from the existing points.

Benefits:

- This would reduce the number of updates that aren't useful
- Locations would converge overtime to the average of reported locations making the location more accurate

Drawbacks:

- It would much harder to change dramatically different GPS information.
- It preferences the first people to enter GPS locations and would need to remove outliers to improve the metric.

Tag Values Normalized

The tag values have a lot of differing values. My profile method shows 862 different tag value possibilities. The tag element can be more standardized which makes it easier to process location information

Benefits

- A more normalized structure makes the data far easier to process
- Anomalies can be found more easily
- Accuracy auditing can be automated by cross referencing the locations against other services.

Cons

- The enforced structures could introduce compromises the contributors may want to avoid
- The variance in tags are essentially just concentrated in a separate tag, not eliminated

Fix Me Tag

While doing address cleaning I discovered a tag named fixme. The tags were highly varied and seem to be notes that require indiviudual attention. To dig deepter I pulled out all of the nodes that contained fixm tag and wrote it out ot a file.

```
<tag k="fixme" v="Need to copy relations/tags from northbound side"/>
<tag k="fixme" v="check speed limits and route number (434 or 423?)"/>
<tag k="fixme" v="name?"/>
```

Benefits

- This gives ready made issues to fix
- These data points are obviously ones that need to be addressed
- Pre dientified points of data that needs to be cleaned.

Cons

- A lot of the data points are GPS Issues
- The Data points don't seem to be uniform so they'll need to be procssed manually
- It's uknown who's making these comments and how they arrived to this conclusion