OpenStreetMap Data Wrangling with MongoDB

Riya Mokashi

The file can be found at: https://mapzen.com/data/metro-extracts/your-extracts/52e5d6b2de8d

The location I chose for this analysis was the Duluth-Superior area. Superior is my hometown, however due to size and location, Duluth and Superior often get lumped together although they exist over seperate city lines. As of personal experience I can say that these two cities are very small, and extremely localized, so I am going into this analysis extremely interested in what will be found through the openstreetmap analysis.

In [25]:

```
import sys
sys.path.append('./data_analysis_programming')

data_name = "duluthsuperior.osm_01"
OSMFILE = "{}.osm".format(data_name)
```

Problems in the Analysis

Tag key name problems

So I first went about and found any tags that had problems or problem names. These are listed as:

- "lower", for tags that contain only lowercase letters and are valid,
- "upper", for tags that contain upercase letters,
- "lower_colon", for otherwise valid tags with a colon in their names,
- "upper_colon", for tags contain both uppercase strings delimited by a colon,
- "multiple_colons", for tags contain more than two colon seperated strings,
- "numbers", for tags that contain a digit, and
- "problemchars", for tags with problematic characters, and
- "other", for other tags that do not fall into the other three categories.

In [26]:

```
# Find problems with tag names
import tags as tags_processor
tag_problems = tags_processor.process_map(OSMFILE)
print "The number of keys in each of the 'problem' categories:"
print tag_problems['counts']
unique_key_names = tags_processor.unique_tag_keys(OSMFILE)
print "There are {} unique tag key names in the data set.".format(len(unique_key_names))
```

```
The number of keys in each of the 'problem' categories:
{'problemchars': 1, 'upper': 169, 'lower': 53080, 'upper_colon': 441, 'numbers': 1904,
'multiple_colons': 131, 'lower_colon': 38315, 'other': 12}
There are 609 unique tag key names in the data set.
```

Now that we know what kind of problematic keys are at hand, we can further analyze them. Here we will be taking an especially close look at the problemchars keys and the other keys.

In [27]:

```
tag_problems['keys']['problemchars']
```

Out[27]:

```
['Hawk Ridge Observatory']
```

So this is a famous observatory in the Duluth-Superior region, but the program is having a hard time reading this because of the

spaces in between the words. For this we will be replacing the spaces - " " with underscores which the program can understand "_".

In [28]:

```
set(tag_problems['keys']['other'])
```

Out[28]:

```
{'name:bat-smg',
 'name:be-tarask',
 'name:cbk-zam',
 'name:fiu-vro',
 'name:map-bms',
 'name:nds-nl',
 'name:roa-tara',
 'name:zh-classical',
 'name:zh-min-nan',
 'name:zh-yue',
 'voltage-high'}
```

These were likely categorized here because of the hyphens, however this should be okay, so we won't try to further refine this.

Street name problems

In [29]:

```
# Find problems with street names
import audit as street_name_auditor
street_name_auditor.audit(OSMFILE)
```

Out[29]:

```
defaultdict(set,
            {'2': {'East US Highway 2'},
             '35': {'State Highway 35'},
             'Ave': {'Junction Ave'},
             'B': {'County Road B'},
             'Circle': {'Oakland Circle', 'Taylor Circle'},
             'Dr': {'Lake Place Dr', 'Mall Dr'},
             'E': {'E Avenue E'},
             'East': {'13th Avenue East',
              '21st Avenue East',
              '23rd Avenue East',
              '26th Avenue East',
              '2nd Avenue East',
              '3rd Avenue East',
              'South 32ed Avenue East'},
             'Entrance': {'West Central Entrance'},
             'F': {'County Road F'},
             'Highway': {'Miller Trunk Highway'},
             'Rd': {'Haines Rd', 'S Midbon Rd', 'W Arrowhead Rd'},
             'South': {'Lake Avenue South'},
             'St': {'W 2nd St',
              'W 3rd St',
              'W Superior St',
              'West 1st St',
              'West First St'},
             'West': {'South 18th Avenue West', 'South 57th Avenue West'}})
```

So this list of streets and roads is very telling that this data is largely incomplete. Nevertheless we can still analyze and clean whatever we have. In the audit.py file further programming has been included that expands abbreviations such as Rd to Road or St to Street. On the other hand I don't see any incorrect data, outside of "32ed Avenue East" which has also been fixed, so there is no further clarification required.

Load the Data

In [30]:

```
import data as data_processor
data = data_processor.process_map(OSMFILE, True)
```

```
data = data_processor.process_map(OSMFILE, True)

from pymongo import MongoClient
client = MongoClient("mongodb://localhost:27017")
db = client.minneapolis_sample
collection = getattr(db, data_name)

collection_count = collection.count()

# If the collection size dowsn't match the data size, load/reload the data.
if collection_count != len(data):
    if collection_count > 0:
        collection.drop()
    collection.insert_many(data)

collection_count = collection.count()
sample_record = collection.find_one()
print "Number of records in the {} MongoDB collection: {}".format(data_name, collection_count)
print "A sample record from the {} MongoDB collection: {}".format(data_name, sample_record)
```

```
Number of records in the duluthsuperior.osm_01 MongoDB collection: 260900
A sample record from the duluthsuperior.osm_01 MongoDB collection: {u'name': u'Duluth',
u'created': {u'changeset': u'43780386', u'version': u'14', u'uid': u'145231', u'timestamp':
u'2016-11-19T03:04:31Z', u'user': u'woodpeck_repair'}, u'is_in': u'Minnesota USA', u'wikipedia': u
'en:Duluth, Minnesota', u'pos': [46.7729322, -92.1251218], u'visible': None, u'place': u'city', u'
_id': ObjectId('588eb2a4b4b2d42114a59181'), u'type': u'node', u'id': u'19188464', u'population': u
'86265'}
```

Overview of the data

In [31]:

```
# Size of the OSM and JSON files.
import os
def get_size_in_mb_of_relative_file(file_name):
    wd = %pwd
    return os.stat(wd + '/' + file_name).st_size / 1000.0 / 1000.0

for file_name in [OSMFILE, OSMFILE + '.json']:
    file_size = get_size_in_mb_of_relative_file(file_name)
    print "{} is {} MB in size.".format(file_name, file_size)
```

```
duluthsuperior.osm_01.osm is 50.864712 MB in size.
duluthsuperior.osm_01.osm.json is 82.758683 MB in size.
```

The entirety of the Duluth/Superior area just barely scrapes the 50 mb minimum.

In [32]:

```
# Number of records:
print "Number of records in the {} MongoDB collection: {}".format(data_name, collection_count)
# Number of nodes:
num_nodes = collection.find({"type":"node"}).count()
print "Number of 'node' records in the {} MongoDB collection: {}".format(data_name, num_nodes)
# Number of ways:
num_ways = collection.find({"type":"way"}).count()
print "Number of 'way' records in the {} MongoDB collection: {}".format(data_name, num_ways)
# Number of unique users:
num_unique_users = len(collection.distinct("created.user"))
print "Number of unique 'users' in the {} MongoDB collection: {}".format(data_name,
num_unique_users)
# Number of unique amenity types:
num_unique_amenities = len(collection.distinct("amenity"))
print "Number of unique 'amenity' values in the {} MongoDB collection: {}".format(data_name, num_u
nique_amenities)
```

```
Number of records in the duluthsuperior.osm_01 MongoDB collection: 260900
Number of 'node' records in the duluthsuperior.osm_01 MongoDB collection: 238815
Number of 'way' records in the duluthsuperior.osm_01 MongoDB collection: 22081
Number of unique 'users' in the duluthsuperior.osm_01 MongoDB collection: 242
Number of unique 'amenity' values in the duluthsuperior.osm_01 MongoDB collection: 48
```

Analysis

Number of Universities and Colleges

In [33]:

```
# Number of universities and colleges:
num_colleges = collection.find({"amenity": {"$in": ["university", "college"]}}).count()
print "Number of 'university' and 'college' records in the {} MongoDB collection: {}".format(data_
name, num_colleges)
```

```
Number of 'university' and 'college' records in the duluthsuperior.osm_01 MongoDB collection: 4
```

The Duluth - Superior area hosts very few colleges and universities, once again going back to the extremely local roots at play here. This can certainly be seen through this analysis.

Food Analysis

As a self-proclaimed foodie I was very excited to see how the analysis of this section would play out. This region is rich in local mom and pop restaurants, however I was doubtful that these would appear in the analysis. My hypothesis was unfortunatly proven right.

In [34]:

```
# Number of unique cuisine types:
num_unique_cuisines = len(collection.distinct("cuisine"))
print "Number of unique 'cuisine' values in the {} MongoDB collection: {}".format(data_name, num_u
nique_cuisines)
```

```
Number of unique 'cuisine' values in the duluthsuperior.osm_01 MongoDB collection: 13
```

In [35]:

```
import cuisine as cuisine_auditor
food_nodes = cuisine_auditor.audit(OSMFILE)

food_nodes_with_cuisine_and_amenity = [n for n in food_nodes if 'cuisine' in n and 'amenity' in n]
food_nodes_without_cuisine = [n for n in food_nodes if 'cuisine' not in n]
food_nodes_without_amenity = [n for n in food_nodes if 'amenity' not in n]
print "Number of food nodes: {}".format(len(food_nodes))
print "Number of food nodes with a cuisine and amenity:
{}".format(len(food_nodes_with_cuisine_and_amenity))
print "Number of food nodes without a cuisine: {}".format(len(food_nodes_without_cuisine))
print "Number of food nodes without an amenity: {}".format(len(food_nodes_without_amenity))
```

```
Number of food nodes: 70
Number of food nodes with a cuisine and amenity: 34
Number of food nodes without a cuisine: 36
Number of food nodes without an amenity: 0
```

In [36]:

```
from collections import Counter

amenities = []
for node in food_nodes_without_cuisine:
    amenities.append(node['amenity'])

print "Amenity counts for food nodes without a cuisine:"
print Counter(amenities)
```

```
Amenity counts for food nodes without a cuisine:
Counter({'restaurant': 20, 'fast_food': 12, 'cafe': 4})
```

In [37]:

```
for amenity in cuisine_auditor.food_amenities:
    cuisines = []
    relevant_nodes = [n for n in food_nodes_with_cuisine_and_amenity if n['amenity'] == amenity]
    for node in relevant_nodes:
        cuisines.append(node['cuisine'])
    print "Cuisine counts for {} nodes:".format(amenity)
    print Counter(cuisines)
    print '\n'
```

```
Cuisine counts for restaurant nodes:
Counter({'pizza': 5, 'american': 2, 'italian': 2, 'sandwich': 1, 'mexican': 1, 'fish': 1,
'regional': 1, 'burger': 1, 'snack': 1, 'chicken': 1, 'italian_pizza': 1})


Cuisine counts for cafe nodes:
Counter({'ice_cream': 1, 'coffee_shop': 1})


Cuisine counts for fast_food nodes:
Counter({'burger': 8, 'sandwich': 4, 'mexican': 1, 'pizza': 1})
```

In [38]:

```
from difflib import SequenceMatcher

def similarity_by_name(a, b):
    if 'name' in a and 'name' in b:
        a = a['name'].replace('the', '').lower()
        b = b['name'].replace('the', '').lower()
        return SequenceMatcher(None, a, b).ratio()
    else:
        return 0

subject = food_nodes_without_cuisine[0]

processed_nodes = []
food_nodes_with_same_amenity = [n for n in food_nodes_with_cuisine_and_amenity if n['amenity'] == s
ubject['amenity']]
for node in food_nodes_with_same_amenity:
    if 'name' in node:
        processed_nodes.append({'similarity': similarity_by_name(subject, node), 'node': node})

print "Subject name: {}   Subject amenity: {} \n".format(subject['name'], subject['amenity'])
sorted_results = sorted(processed_nodes, key=lambda k: k['similarity'], reverse=True)
for result in sorted_results[:5]:
    node = result['node']
    score = '%.3f' % result['similarity']
    print "Similarity score: {}   Name: {}   Cuisine: {}".format(score, node['name'], node['cuisine
'])
```

```
Subject name: Red Mug Coffee-Cafe   Subject amenity: cafe

Similarity score: 0.474   Name: Northern Shores Coffee   Cuisine: coffee_shop
```

What we see from this analysis is the following:

There are 70 seperate food nodes, 36 of which do not fall under cuisine. Of the other 34 food nodes that do, these can all be split between 13 sub-cuisine categories.

It was odd for me to see that 20 restaurants fell under food nodes that did not include cuisine. This may have been an error on the users' parts, or perhaps the restaurants just simply did not fall under a certain set of guidelines. Either way I would definetly visit this again in the future and attempt to rectify this anomaly.

Of the cusine categories I decided to analyze the fast food, cafe and restaurant nodes. These were interstingly further split up and although I have lived here for quite a while I was surprised to see we have a restaurant that fits under the fish category! If we were to further simplify this data, we may attempt to take out some of the arbitrary labeling such as "snack" and "regional" I am sure we could group these restaurants under other, more broad, labels.

After this I went to examine some similarity data and it was interesting to find that the Red Mug Cafe had a similarity score of .474

After this I went to examine some similarity data and it was interesting to find that the Red Mug Cafe had a similarity score of 1.1? with the Northern Shores Coffee Shop. I personally love the Red Mug but have never been to the other cafe, however I would be very interested in more closely examining the data similarities between these two.

Top Tens

The top 10 most common amenities

In [39]:

```
results = collection.aggregate([
        {"$match": {"amenity": {"$exists": 1}}},
        {"$group": {"_id": "$amenity", "count": {"$sum": 1}}},
        {"$sort": {"count": -1}},
        {"$limit": 10}
    ])
for result in results:
    print result
```

```
{u'count': 596, u'_id': u'parking'}
{u'count': 46, u'_id': u'school'}
{u'count': 37, u'_id': u'restaurant'}
{u'count': 34, u'_id': u'fuel'}
{u'count': 30, u'_id': u'place_of_worship'}
{u'count': 26, u'_id': u'fast_food'}
{u'count': 13, u'_id': u'bank'}
{u'count': 11, u'_id': u'grave_yard'}
{u'count': 10, u'_id': u'theatre'}
{u'count': 10, u'_id': u'bar'}
```

To me, it seems odd that parking has 596 counts, more than 10 times the next highest tag - schools.

Top 10 most commonly found places to eat

In [40]:

```
results = collection.aggregate([
        {"$match": {"amenity": {"$in": ["restaurant", "cafe", "pub", "bar", "fast_food",
"delicatessen"]}}},
        {"$match": {"name": {"$exists": 1}}},
        {"$group": {
                "_id": "$name",
                "amenity": {"$first": "$amenity"},
                "cuisine": {"$push": "$cuisine"},
                "count": {"$sum": 1}
            }},
        {"$project": {
                "_id": 0,
                "count": 1,
                "name": "$_id",
                "type" : {"$concat": [
                        "$amenity", " - ",
                        {"$ifNull": [{"$arrayElemAt": ["$cuisine", 0 ]}, "unknown cuisine"] }
                    ]}
            }},
        {"$sort": {"count": -1}},
        {"$limit": 10}
    ])

for result in results:
    print "Count {}: {} ({})".format(result['count'], result['name'], result['type'])
```

```
Count 6: Subway (fast_food - sandwich)
Count 5: McDonald's (fast_food - burger)
Count 3: Burger King (fast_food - burger)
Count 2: Arby's (fast_food - unknown cuisine)
Count 2: Culver's (fast_food - burger)
Count 2: Dairy Queen (fast_food - unknown cuisine)
Count 2: Wendy's (fast_food - unknown cuisine)
Count 1: Grandma's Saloon and Grill (restaurant - regional)
```

```
Count 1: Charlie Brown's Bar (bar - unknown cuisine)
Count 1: Dining Center (restaurant - unknown cuisine)
```

As we can see from this analysis, chain restaurants see a high count in this region. The chain restaurants that can be found here are very commercialized ones, something that isn't too surprising considering the rather uniform lifestyle of many citizens of this area.

Of the top ten food places, fast food joints take up the first 7! This is interesting, however not surprising, as I am sure that this would likely be the case in other cities. More metropolitan areas, on the other hand, would likely have more diverse cuisine in the top ten list.

Conclusion

Although I am happy to see that a chunk of the Duluth/Superior data has been uploaded to this database, I am by no means uncertain that this is largely incomplete. There are definitly more than 28 streets, roads, and pathways in these two cities. This may largely be due to the difficulties that come from crowd sourced data. This area saw the contributions of 242 users, a number that seems relatively decent until you compare it to other areas. When I was first playing around with this project I tested out the Minneapolis dataset which showed thousands of user contributions.

Perhaps due to the small size I also saw far less inconsistencies than I had in the Minneapolis data set. This may be due to the extremely localized nature of this area, and the greater specificity with which data contributions may be made available.

Some of the greatest difficulties I had with this project included efficently uploading the data into MongoDB and figuring out what to analyze. I saw some great analysis done on bigger cities, but was disappointed to find that the meager amount of data granted to me did not allow me much to play around with.

I believe a great way to improve this dataset would be to offer a more interesting approach to data collecting. One method that really stuck out to me was the implementation of some form of user data contribution options in popular apps such as Pokemon Go!. It would be something that would not only be passively engaging to users, but it would also lead to a greater increase in localized data.

Benefits of this:

- Localized data
- User friendly
- Likely to be accepted

Hardships:

- Hard to get it implemented
- Could lead to corrupt user data

If I were to do this in the future I would likely look for more unique analysis that can be done with this data and I would attempt to make this information more readily readable. However, I do think this was a good place to go off of for a start.