Data Wrangle OpenStreetMaps Data

by Rica Enriquez, July 1, 2015

In this project, the OpenStreetMap data for Cambridge, United Kingdom is explored. It was downloa
(https://mapzen.com/data/metro-extracts) on July 1, 2015. It was prepared for MongoDB using nan
the 'cambridge' collection.

Import the database for querying below

In [1]:
```
from pymongo import MongoClient
import pprint
import os

client = MongoClient()
db = client["udacity"]
```

Section 1. Problems Encountered in the Map

There were a few problems with the street names. "chieftain" and "sweetpea" were not capitalized.
scheme similar to the "Improving Street Names" script from Lesson 6.11.

When importing the data, all labels were included. Many of these labels were only available for a sm
underutilized upper labels and lower lables are removed. If a subset of the dataset is used in the fut

Removing underutilized labels

Using MongoDB, only upper labels that had at least 1000 documents were kept. The "nanoproject_
looped through in "nanoproject_2_query.py" to check the number of documents it is used in. If there
database is updated.

In [2]:
```
all_labels = ['real_ale', 'fhrs', 'anglican', 'dance', 'sS052', 'demodifie
                  'is_in', 'max_age', 'created_by', 'fax', 'cctv', 'mph',
                  'school', 'level', 'notes', 'bus_stop', 'ncn_1', 'real_c
                  'cost',
                  'exit_to', 'leaf_type', 'access', 'fast_food', 'eligibil
                  'used_to_be', 'permit_holders', 'microbrewery', 'survey'
                  'lwn',
                  'vehicle', 'have_riverbank', 'type', 'start_date', 'entr
                  'campaigned_for_by',
                  'give_way', 'visibility', 'site', 'phone', 'traffic_calm
                  'history', 'estate', 'lock', 'currency', 'ncn_ref', 'por
                  'gate', 'uk_postcode_centroid', 'FIXME', 'description',
                  'rental'
```

```
                    'rental',
                    'natural', 'lcn_ref', 'wheelchair', 'outdoor_seating', '
                    'postal_code', 'motorcycle', 'int_ref', 'pitch', 'covere
                    'material', 'foot', 'tourism', 'smoothness', 'fixme', 'n
                    'embankment',
                    'crossing', 'kerb', 'name_1', 'frequency', 'naptan', 'ac
                    'highways_agency', 'ref', 'brewery', 'highway', 'barrier
                    'electrified',
                    'was_called', 'old_amenity', 'accommodation', 'tenant',
                    'box_type', 'turn', 'place', 'high_capacity', 'support',
                    'priority',
                    'motorcar', 'park_ride', 'enforcement', 'noname', 'est_w
                    'population', 'multi_storey', 'royal_cypher', 'aeroway',
                    'occupier', 'nqa', 'sidewalk', 'hgv', 'lit', 'takeaway',
                    'aerodrome', 'url', 'medical', 'tactile_paving', 'shop',
                    'last_survey', 'gauge', 'mapillary', 'wood', 'fuel', 'ia
                    'artist', 'motorboat', 'public_transport', 'power_source
                    'lamp_type',
                    'route_ref', 'parking', 'sport', 'power_supply', 'capaci
                    'boundary',
                    'email', 'screen', 'denomination', 'key', 'substation',
                    'genus', 'comment', 'maintainer', 'wall', 'loading_gauge
                    'ele', 'alt_description', 'boat', 'speech_output', 'mkgm
                    'direction', 'lanes', 'building_1', 'craft', 'official_n
                    'parking_space', 'replaced', 'overtaking', 'layer', 'ons
                    'guided_busway', 'beer_garden', 'waterway', 'cuisine', '
                    'collection_times',
                    'status', 'wires', 'cyclestreets_id', 'fence_type', 'fru
                    'oneway',
                    'landmark', 'left', 'taxi', 'livestock', 'proposed', 'ho
                    'guest_house',
                    'isced', 'toilets', 'generator', 'TODO', 'bench', 'sourc
                    'historic',
                    'lcn', 'psv', 'furniture', 'vending', 'tower', 'internet
                    'local_ref', 'man_made', 'religion', 'artwork_type', 'po
                    'footway',
                    'industry', 'taxon', 'supervised', 'step_count', 'female
                    'opening_hours',
                    'museum', 'width', 'occupier3', 'occupier2', 'admin_leve
                    'construction',
                    'diaper', 'courts', 'old_name', 'real_fire', 'circuits',
                    'crossing_ref',
                    'cinema', 'carriageway_ref', 'maxheight', 'cafe', 'cable
                    'interior_decoration', 'cycleway', 'department', 'denota
                    'diet',
                    'min_age', 'maxstay', 'opened', 'building', 'yelp', 'wif

# Number of documents in the collection
N = db.cambridge.find().count()

removed = []
kept = []

# Remove labels used in less than 1000 documbets
```

```
for label in all_labels:
    pipeline = [{"$group": {"_id": "$" + label, "count": {"$sum": 1}}}, {"
    result = list(db.cambridge.aggregate(pipeline))
    if len(result) > 0:
        n = result[0]["count"]
        if n >= N - 1000:
            db.cambridge.update({}, {"$unset": {label: ""}}, multi=True)
            removed.append(label)
        else:
            kept.append(label)
print len(removed), "labels were removed and", len(all_labels) - len(remov
print "The labels kept are:", kept
```

```
336 labels were removed and 11 labels were kept.
The labels kept are: ['address', 'amenity', 'entrance', 'natural', 'foot',
rator']
```

Removing underutilized sublabels

Similarly, only lower labels that had at least 500 documents were kept. The "kept" list from the Mon
dictionary of the list of kept upper labels and their sublabels. This dictionary is then looped through
each sublabel is used in, using MongoDB. If there are less than 500 documents, that sublabel is rem
contain any sublabels, it is also removed. The final structure of the collection is printed out.

In [3]:
```
kept_sublabels = {'building': ['name', 'level', 'levels', 'min_level', 'ma
                  'maxspeed': ['type', 'ype'],
                  'name': ['cy', 'eo', 'ru', 'sr', 'uk', 'zh', 'en', 'zh_p
                  'service': ['bicycle:pump', 'bicycle:chain_tool'], 'acce
                  'source': ['crossing', 'addr', 'name', 'phone', 'populat
                             'location', 'start_date', 'fhrs:id', 'opening
                             'ele', 'cost', 'taxon', 'database', 'position
                             'traffic_calming', 'designation', 'operator',
                             'bicycle:backward', 'tourist_bus:backward', '
                             'hgv', 'outline', 'maxspeed:date', 'addr:hous
                             'bridge', 'addr:postcode', 'tracktype', 'heig
                  'address': ['street', 'postcode', 'housenumber', 'housen
                              'interpolation', 'flat', 'flats', 'place', '
                  'ref': ['university_of_cambridge', 'observado']}

# Remove sublabels used in less than 500 documents
removed_sub = {}
kept_sub = {}
for label in kept_sublabels.keys():
    for sublabel in kept_sublabels[label]:
        pipeline = [{"$group": {"_id": "".join(["$", label, ".", sublabel]
                    {"$match": {"_id": None}}]
        result = list(db.cambridge.aggregate(pipeline))
        if len(result) > 0:
            n = result[0]["count"]
            if n >= N - 500:
                db.cambridge.update({}, {"$unset": {"".join([label, '.', s
```

```
                try:
                    removed_sub[label].append(sublabel)
                except:
                    removed_sub[label] = [sublabel]
            else:
                try:
                    kept_sub[label].append(sublabel)
                except:
                    kept_sub[label] = [sublabel]

# Remove the upper labels that no longer have sublabels
for label in removed_sub.keys():
    if label not in kept_sub.keys():
        db.cambridge.update({}, {"$unset": {label: ""}}, multi=True)

# Print the final structure of the collection
final_labels = {}
for label in kept:
    if label in kept_sub:
        final_labels[label] = kept_sub[label]
    elif label not in removed_sub:
        final_labels[label] = None

print "The final structure of the collection is:"
pprint.pprint(final_labels)
```

```
The final structure of the collection is:
{'address': ['street',
             'postcode',
             'housenumber',
             'housename',
             'city',
             'country',
             'interpolation'],
 'amenity': None,
 'barrier': None,
 'entrance': None,
 'foot': None,
 'highway': None,
 'landuse': None,
 'natural': None,
 'operator': None,
 'source': ['name']}
```

Remove postcodes that do not start with "CB"

All postcodes in Cambridge start with CB. Listing the postcodes in the collection, documents with a
since it is for a place in Royston and the Stevenage postcode area. Since there are only two docum
approach to extend the collection the surrounding area. Additionally the postcode "CB1" is incompl
Documents with this are also removed.

In [4]:
```
db.cambridge.remove({"address.postcode": "CB1"})
db.cambridge.remove({"address.postcode": "SG8 5TF"})
```

Out[4]:
```
{u'n': 0, u'ok': 1}
```

Update Cities to Cambridge

Some cities were "cambridge" and not "Cambridge", overspecified to "Girton" or "South Cambridg
show that each entry is in Cambridge. Therefore, the "city" is updated to "Cambridge".

In [5]:
```
db.cambridge.update({"address.city": "cambridge"}, {"$set": {"address.city
                     multi=True)
db.cambridge.update({"address.city": "South Cambridgeshire"}, {"$set": {"a
                     multi=True)
db.cambridge.update({"address.city": "Girton"}, {"$set": {"address.city":
db.cambridge.update({"address.city": "11"}, {"$set": {"address.city": "Cam
```

Out[5]:
```
{u'n': 0, u'nModified': 0, u'ok': 1, 'updatedExisting': False}
```

Pare down barrier, entrance, highway, landuse, and operato

Some entries for these sublabels were the same, but in a different format. They were updated to be

In [6]:
```
# Pare down barrier
pipeline = [{"$group": {"_id": "$barrier", "count": {"$sum": 1}}},
            {"$sort": {"_id": -1}}]
result = list(db.cambridge.aggregate(pipeline))
pprint.pprint(result)

db.cambridge.update({"barrier": "fence;wall"}, {"$set": {"barrier": "fence
db.cambridge.update({"barrier": "fence;wall"}, {"$set": {"barrier": "fence
db.cambridge.update({"barrier": "fedr"}, {"$set": {"barrier": None}}, upse
db.cambridge.update({"barrier": "bollards"}, {"$set": {"barrier": "bollard

# Pare down entrance
pipeline = [{"$group": {"_id": "$entrance", "count": {"$sum": 1}}},
            {"$sort": {"_id": -1}}]
result = list(db.cambridge.aggregate(pipeline))
pprint.pprint(result)

db.cambridge.update({"entrance": "secondary_entrance"}, {"$set": {"entranc
                     multi=True)
db.cambridge.update({"entrance": "main_entrance; porters"}, {"$set": {"ent
                     upsert=False, multi=True)
db.cambridge.update({"entrance": "porters;main_entrance"}, {"$set": {"entr
                     upsert=False, multi=True)
db.cambridge.update({"entrance": "main"}, {"$set": {"entrance": "main_entr
db.cambridge.update({"entrance": "emegency"}, {"$set": {"entrance": "emerg
db.cambridge.update({"entrance": "main_entrance;porters;"}, {"$set": {"ent
                     upsert=False, multi=True)
```

```
                    upsert=False, multi=True)

# Pare down highway
pipeline = [{"$group": {"_id": "$highway", "count": {"$sum": 1}}},
            {"$sort": {"_id": -1}}]
result = list(db.cambridge.aggregate(pipeline))
pprint.pprint(result)

db.cambridge.update({"highway": "bus_stand"}, {"$set": {"highway": "bus_st

# Pare down landuse
pipeline = [{"$group": {"_id": "$barrier", "count": {"$sum": 1}}},
            {"$sort": {"_id": -1}}]
result = list(db.cambridge.aggregate(pipeline))
pprint.pprint(result)
db.cambridge.update({"landuse": "institututional"}, {"$set": {"landuse": "
                    multi=True)

# Pare down operator
pipeline = [{"$group": {"_id": "$operator", "count": {"$sum": 1}}},
            {"$sort": {"_id": -1}}]
result = list(db.cambridge.aggregate(pipeline))
pprint.pprint(result)
db.cambridge.update({"operator": "YourSpace"}, {"$set": {"operator": "Your
                    multi=True)
db.cambridge.update({"operator": "Your Space"}, {"$set": {"operator": "You
                    multi=True)
db.cambridge.update({"operator": "Trinity College"},
                    {"$set": {"operator": "Trinity College (University of
db.cambridge.update({"operator": "St John's College"},
                    {"$set": {"operator": "St John's College (University c
db.cambridge.update({"operator": "Lucy Cavendish College"},
                    {"$set": {"operator": "Lucy Cavendish College (Univers
                    multi=True)
db.cambridge.update({"operator": "Lloyds"}, {"$set": {"operator": "Lloyds
db.cambridge.update({"operator": "King's College"},
                    {"$set": {"operator": "King's College (University of C
db.cambridge.update({"operator": "King's College (University Of Cambridge)
                    {"$set": {"operator": "King's College (University of C
db.cambridge.update({"operator": "Needham Institute"}, {"$set": {"operator
                    upsert=False, multi=True)
db.cambridge.update({"operator": "Gonville and Caius College (University c
                    {"$set": {"operator": "Gonville & Caius College (Unive
                    multi=True)
db.cambridge.update({"operator": "EDF"}, {"$set": {"operator": "EDF Energy
db.cambridge.update({"operator": "Clare College"},
                    {"$set": {"operator": "Clare College (University of Ca
db.cambridge.update({"operator": "Christ's College"},
                    {"$set": {"operator": "Christ's College (University of
{u'_id': u'kissing_gate', u'count': 12},
{u'_id': u'kerb', u'count': 34},
{u'_id': u'height_restrictor', u'count': 1},
{u'_id': u'hedge', u'count': 490},
{u'_id': u'gravestone', u'count': 1},
{u'_id': u'gate', u'count': 807}
```

```
 {u'_id': u'gate', u'count': 807},
 {u'_id': u'full-height_turnstile', u'count': 1},
 {u'_id': u'fence', u'count': 5405},
 {u'_id': u'entrance', u'count': 155},
 {u'_id': u'ditch', u'count': 1},
 {u'_id': u'cycle_barrier', u'count': 39},
 {u'_id': u'chicane', u'count': 2},
 {u'_id': u'cattle_grid', u'count': 36},
 {u'_id': u'car_trap', u'count': 9},
 {u'_id': u'boom', u'count': 1},
 {u'_id': u'bollard', u'count': 186},
 {u'_id': u'block', u'count': 1},
 {u'_id': u'barrier', u'count': 3},
 {u'_id': None, u'count': 298180}]
[{u'_id': u'Your Space Apartments', u'count': 5},
```

Section 2. Overview of the Data

A statistical overview of the dataset with the MongDB queries used to obtain such statistics are bel

In [7]:
```
print "The size of 'cambridge_england.osm' is", os.stat("cambridge_englanc
print "The size of 'cambridge_england.osm.json' is", os.stat("cambridge_er
N2 = db.cambridge.find().count()
print "There are", N, "documents in the original set and", N2, "documents
pipeline = [{"$group": {"_id": "$created.user", "count": {"$sum": 1}}}]
print "There are", len(list(db.cambridge.aggregate(pipeline))), "unique us
print "There are", db.cambridge.find({"type": "node"}).count(), "nodes."
print "There are", db.cambridge.find({"type": "way"}).count(), "ways."
pipeline = [{"$group": {"_id": "$created.user", "count": {"$sum": 1}}},
            {"$sort": {"count": -1}},
            {"$limit": 1}]
print list(db.cambridge.aggregate(pipeline))[0]['_id'], "contributed the m
    list(db.cambridge.aggregate(pipeline))[0]['count'], "documents."
pipeline = [{"$group": {"_id": "$created.user", "count": {"$sum": 1}}},
            {"$group": {"_id": "$count", "num_users": {"$sum": 1}}},
            {"$sort": {"_id": 1}},
            {"$limit": 1}]
print list(db.cambridge.aggregate(pipeline))[0]['num_users'], "users contr
pipeline = [{"$group": {"_id": "$amenity", "count": {"$sum": 1}}},
            {"$match": {"_id": {"$ne": None}}},
            {"$sort": {"count": -1}},
            {"$limit": 5}]
print list(db.cambridge.aggregate(pipeline))[0]["_id"], list(db.cambridge.
    "_id"], "are the top two amenities."
```

```
The size of 'cambridge_england.osm' is 61.828774 MB.
The size of 'cambridge_england.osm.json' is 87.071371 MB.
There are 306428 documents in the original set and 306428 documents in the
There are 453 unique users.
There are 257058 nodes.
There are 49351 ways.
smb1001 contributed the most to this collection with 81443 documents.
120 users contributed once.
university bicycle_parking are the top two amenities.
```

Section 3. Additional Ideas

Some addresses have house names. It would be interesting to know if there's a certain postal code
Additionally, it would be interesting where the top amenities are located.

In [8]:
```
# Find the top postcodes with housenames
pipeline = [{"$match": {"address.housename": {"$exists": True}}},
                {"$group": {"_id": "$address.postcode", "count": {"$sum":
                {"$sort": {"count": -1}}]
pprint.pprint(list(db.cambridge.aggregate(pipeline)))

# Find the top operators with housenames
pipeline = [{"$match": {"address.housename": {"$exists": True}}},
                {"$group": {"_id": "$operator", "count": {"$sum": 1}}},
                {"$sort": {"count": -1}}]
pprint.pprint(list(db.cambridge.aggregate(pipeline)))

# Find the top amenties with postcodes and sort by postcode
pipeline = [{"$match": {"amenity": {"$ne": None}}},
            {"$match": {"amenity": {"$ne": "university"}}},
            {"$match": {"address.postcode": {"$ne": None}}},
            {"$group": {"_id": {"amenity": "$amenity",
                                "postcode": "$address.postcode"},
                        "count": {"$sum": 1}}},
            {"$sort": {"count": -1}},
            {"$group": {"_id": "$_id.amenity",
                        "info": {"$push": {
                            "postcode": "$_id.postcode",
                            "count": "$count"},},
                        "count": { "$sum": "$count"}}},
            {"$sort": {"count": -1}},
            {"$limit": 5}]
pprint.pprint(list(db.cambridge.aggregate(pipeline)))
```

```
 {u'_id': u'CB3 0JB', u'count': 1},
 {u'_id': u'CB2 1UA', u'count': 1},
 {u'_id': u'CB4 1HQ', u'count': 1},
 {u'_id': u'CB2 3LL', u'count': 1}]
[{u'_id': None, u'count': 514},
 {u'_id': u'University of Cambridge', u'count': 53},
 {u'_id': u'Clare College (University of Cambridge)', u'count': 6},
 {u'_id': u'Selwyn College (University of Cambridge)', u'count': 4},
 {u'_id': u'Anglia Ruskin University', u'count': 4},
 {u'_id': u'Riverside ECHG', u'count': 3},
 {u'_id': u'CRM Students', u'count': 2},
 {u'_id': u'Murray Edwards College (University of Cambridge)', u'count': 2
 {u'_id': u'Newnham College (University of Cambridge)', u'count': 2},
 {u'_id': u'Girton College (University of Cambridge)', u'count': 2},
 {u'_id': u"Christ's College (University of Cambridge)", u'count': 2},
 {u'_id': u'British Antarctic Survey', u'count': 1},
 {u'_id': u'Clare Hall (University of Cambridge)', u'count': 1},
 {u'_id': u"St John's College (University of Cambridge)", u'count': 1},
 {u'_id': u'Evangelical Lutheran Church of England', u'count': 1},
```

Section 4. Conclusions

The Cambridge, England OpenStreetMap dataset is full of information. However, this can also be cu
labels and sublables were removed. It may be useful to move the information to an exisiting label ra
"have_riverbank" and "trees" can be moved to the kept "natural" label. Additionally, there are some
"cuisine" was prematurely removed. The top cuisine for this area couldn't be examined with Mongo